FACULTAD DE INFORMÁTICA
UNIVERSIDAD POLITÉCNICA DE MADRID

UNIVERSIDAD POLITÉCNICA DE MADRID

FACULTAD DE INFORMÁTICA

TRABAJO FIN DE CARRERA

Aplicación GeoBuddies para Android

AUTOR: Mónica Batanero La Rotta

TUTOR: Oscar Corcho García

Agradecimientos

Agradecer a Oscar Corcho los consejos y la ayuda prestada durante la realización de este proyecto.

A mis padres, por su apoyo durante estos años.

Dedicar especialmente a mi pareja, David, cuyo respaldo y ánimos han sido decisivos en momentos de angustia y desesperación.

En definitiva, agradecer a todas aquellas personas que en mayor o menor medida han ayudado a que este proyecto se desarrollase.

Índice General

1. Introducción

El ser humano es una especie social por naturaleza. Como tal, necesita del afecto de otras personas así como de la existencia de comunicación. Siendo este último, quizás, uno de los aspectos que más ha marcado la evolución de las tecnologías en los últimos años.

Analizando detenidamente el desarrollo de la tecnología durante el siglo XX, es posible distinguir que el uso destinado para gran parte de estas tecnologías no es otro que el de proporcionar información a las personas, ya sea unilateral o bilateralmente. Se invirtieron grandes cantidades de dinero en investigación lo que, unido a las guerras o a la amenaza de posibles guerras, contribuyó al avance de la ciencia y la tecnología modernas a un ritmo vertiginoso. Por ello, hubo muchos avances en el campo militar (la radio, el rádar, la grabación de sonido…), que fueron claves para invenciones como el teléfono, el fax o el almacenamiento de datos.

Ya desde un primer momento, Internet supuso una auténtica revolución en cuanto a la forma que tenían las personas de comunicarse unas con otras, así como el gran impacto que supone que millones de personas tengan acceso inmediato a una extensa y variada cantidad de información.

Una vez se popularizó el uso de Internet en los países más desarrollados (en la década de los 90), el principal objetivo era el de poder realizar búsquedas concretas y obtener la información relacionada a esas búsquedas de forma sencilla y rápida.

Seguidamente, se fueron proponiendo servicios que atendían a las necesidades del gran número de personas que se incorporaban al fenómeno de la red (correo electrónico, foros de discusión, chats, servicios de compartición de información, …).

Todos los avances referentes a este tipo de servicios se han desarrollado basándose en la conducta del ser humano, que, como ya se ha mencionado anteriormente, posee una alta necesidad de relacionarse. Es por ello que las personas desean mantener el contacto con aquéllas que le son afines en uno u otro aspecto, sin importar su

ubicación física. Esta tarea se ha visto facilitada en la actualidad por los que hoy se conocen como servicios de redes sociales.

Los servicios de redes sociales han supuesto una nueva forma de relacionarse y han desatado la euforia entre los más jóvenes, llegando a formar verdaderos grupos de "amigos virtuales" sin la existencia de ningún tipo de barrera física, y sin la limitación de contactar únicamente con personas cercanas o conocidas.

Sin embargo, no debemos olvidar las altas posibilidades de mercado que tienen estas redes para numerosas empresas, que ven en ellas un nuevo mundo comercial y de alto impacto personal. Se puede tomar como ejemplo el hecho de que, desde un par de años atrás, se han llevado a cabo grandes esfuerzos para integrar el uso de las redes sociales en terminales móviles, proporcionándole al usuario un contacto permanente con lo que ocurre en su entorno y en la sociedad.

Uno de los usos más importantes que se le da a la red es el de compartir experiencias y emociones, tanto con aquellas personas que conocemos y nos importan, como con aquéllas que nos son desconocidas pero a las que se les puede ayudar en sus decisiones. No existe persona que disponga de acceso a Internet que no compare y contraste elementos (hoteles, espectáculos, vehículos, …) antes de decantarse por uno en particular. Nos basamos en las experiencias de otras personas para formar nuestra propia opinión subjetiva y, en base a ella, tomar una decisión final. Si algo nos gusta, lo contamos en la red (a través de foros, redes sociales, páginas Web de opinión específicas según el servicio, …); pero si algo no nos gusta, también lo plasmamos para que nadie cometa el mismo error que nosotros.

Gracias a la evolución del hardware, la interrelación entre estas tecnologías ha propiciado la demanda de aplicaciones que faciliten la realización de acciones como pueden ser enviar un correo, un archivo o escribir un mensaje en nuestro servicio de red social, desde nuestro terminal móvil, sin la necesidad de usar un ordenador convencional; así como aquéllas que nos permiten recomendar (o no) un determinado servicio apoyando nuestro argumento con fotos, vídeos y hasta localización geográfica.

Este proyecto pone de manifiesto la importancia que han adquirido las aplicaciones móviles que permiten al usuario llevar a cabo estas tareas, tanto por portabilidad como por facilidad de uso.

Coincidiendo con el año santo del Xacobeo, y aprovechando el auge de las redes sociales, se ha querido implementar una aplicación que permitiese a los peregrinos del Camino de Santiago compartir vivencias y opiniones sobre los distintos lugares que conforman las rutas. Se pretende darles la posibilidad de recomendar parajes, pueblos, restaurantes, alojamientos, …; en resumen, información sobre todos aquellos aspectos que pueden serles de interés.

Teniendo clara la finalidad que se le quería dar a la aplicación, era necesario elegir una plataforma de desarrollo adecuada. Dada la reciente aparición del sistema operativo *Android* y su rápida expansión en el mercado, acogido por numerosas marcas de terminales móviles y un alto número de usuarios, pareció una gran oportunidad para explorar su kit de desarrollo y las posibilidades que ofrecía. Además, fue un factor importante para tomar esta decisión el hecho de que sea un sistema operativo abierto, lo que permite al desarrollador llevar a cabo sus ideas y poder usarlas de primera mano en su dispositivo, así como ofrecerlas al resto de usuarios de la comunidad *Android* fácilmente.

La *SDK* que ofrece esta plataforma es muy completa y, junto al emulador y la amplia documentación existente, facilita enormemente la tarea del desarrollador en cuanto a implementación y pruebas.

Centrándonos un poco más en la aplicación que se ha buscado desarrollar en este proyecto, pasamos a presentar los problemas que se han pretendido resolver hasta cómo se han resuelto.

Si se profundiza en las necesidades de información que tiene un peregrino del Camino de Santiago una vez que se encuentra en la ruta, obtendremos las siguientes:

- Trazado y perfil de cada etapa
- Lugares de interés cultural
- Hospedaje
- Restauración
- Otros servicios (atención médica, ocio, etc.)

Por lo tanto, el objetivo principal de la aplicación es poder proveer al peregrino de toda esta información de primera mano. La mejor manera de conseguir esto es creando una comunidad de peregrinos, unidos por un mismo objetivo, que compartan sus experiencias en el camino. Gracias a las posibilidades que ofrece el entorno de desarrollo de *Android*, un usuario de la aplicación podrá apoyar sus comentarios con fotos, vídeos y mapas de rutas que hayan seguido.

Como ya se ha mencionado, la tecnología empleada para llevar a cabo el desarrollo de la aplicación es la plataforma *Android*, cuyas características más importantes se detallan a continuación:

- Sistema operativo orientado a dispositivos móviles y cuyo código fuente está disponible bajo licencias de software libre y código abierto.
- Permite el desarrollo de terceros a través de su *SDK* y mediante el lenguaje de programación Java.
- Permite el almacenamiento de datos estructurados mediante SQLite.
- Proporciona soporte para medios con formatos comunes de audio, vídeo e imágenes.
- Facilita la integración del hardware en la aplicación (Bluetooth, GPS, cámara de fotos, brújula, …).
- Integración con Google Maps.
- Emulador incluido en el entorno de desarrollo.

Además, facilita que los desarrolladores compartan sus aplicaciones, de manera gratuita o de pago, a través del *Android Market*, al que tienen acceso los teléfonos con dicho sistema operativo.

2. Estado del Arte y Aplicación GeoBuddies Existente

2.1 Estado del Arte de la Web 2.0

2.1.1 ¿Qué es la Web 2.0?

La *Web 2.0* se asocia con la evolución de las aplicaciones tradicionales hacia aplicaciones Web enfocadas al usuario final, facilitando la compartición de información, la interoperatividad entre sistemas heterogéneos, el diseño centrado en el usuario y la colaboración.

¿CUÁNDO SURGE LA WEB 2.0?

Su antecesor, la *Web 1.0*, hacía referencia a la comunidad de páginas Web estáticas programadas en HTML (HyperText Mark-up Language) que no se actualizaban frecuentemente. Más tarde surgió la Web 1.5, donde los gestores de contenidos (CMS – Content Mangement Systems) servían páginas HTML dinámicas cuya actualización era más fácil y cómoda gracias al almacenamiento de la información en bases de datos.

El fenómeno Web 2.0 surge hacia el 2003, cuando se empieza a buscar que las aplicaciones se orienten a la interacción de los usuarios y a las redes sociales. El término fue acuñado por Dale Dougherty, de O'Reilly, durante una tormenta de ideas llevada a cabo con MediaLive International para una conferencia. Fue en octubre de 2004 cuando se lanzó la primera de ellas, en la que se sugirió el hecho de que la Web estaba cambiando y su modelo de negocio evolucionando.

Por tanto, es posible definir la Web 2.0 como el conjunto de sitios Web, utilidades y servicios interactivos, que se sustentan sobre una base de datos, en los que los usuarios del servicio son, junto con la información, la pieza central pues, sin ellos, "dejarían de existir". Son los encargados de modificar esas bases de datos tanto en contenido como en formato de presentación.

Algunos ejemplos de la Web 2.0 son:

- Servicios Web: conjunto de estándares y protocolos que permiten a aplicaciones desarrolladas bajo distintas plataformas y en distintos lenguajes compartir información entre sí (aportan interoperabilidad).
- Aplicaciones Web: aplicaciones a las que los usuarios pueden acceder desde un servidor Web mediante un navegador.
- Servicios de redes sociales: basados en la Web y que ofrecen una serie de alternativas para que los usuarios puedan interactuar entre sí.
- Bitácoras: sitio Web actualizado periódicamente por su administrador y que recopila artículos o textos de forma cronológica.

En la siguiente página se muestran estos y otros ejemplos sobre un mapa (presentado en el libro "La Web 2.0", publicado por la Fundación Orange) en el que se recogen los principales conceptos que se tienden a relacionar con la Web 2.0.

2.1.2 Principios de la Web 2.0[1]

Todas las aplicaciones y sitios que surgieron a principios de la década del 2000 estaban orientados a las personas, con funcionalidades y características muy interesantes, ya que nunca se habían llevado a cabo proyectos semejantes. Es por esto que este fenómeno hace referencia a una Web más social que comercial, a una mejora de la "World Wide Web". En resumen, enfatiza aquellas plataformas y herramientas que permiten al usuario "comentar", "publicar", "modificar", "etiquetar", etc.

Se podría decir que un sitio o aplicación englobada dentro de la etiqueta "Web 2.0" debe cumplir los siguientes principios:

- No existen productos, si no *servicios*.
- Toma la World Wide Web como plataforma: implementación de aplicaciones embebidas en la Web.
- Personalización: al ser único cada individuo, no se puede esperar que todos deseen las mismas cosas, de la misma forma y en el mismo momento. Ésta es

[1] Fuentes:
- http://gonzbuk.com/2007/11/06/publicidad-internet-vigo-galicia-adwords-ourense-pontevedra-web-20-web-10/
- http://www.maestrosdelweb.com/editorial/web2/

la razón por la que se debe permitir al usuario *elegir* el qué, cómo y cuándo desea usar una determinada aplicación.

Ejemplos: iGoogle, MySpace, extensiones del Firefox, etc.

- Aprovechar la inteligencia colectiva: la información es lo que mueve Internet y la contribución de los usuarios lo que añade valor a la era Web 2.0. Muchos sitios se diseñan con el fin de promover la participación. Ejemplos:
  - Wikipedia – permite añadir, eliminar o modificar contenidos por cualquier usuario de la red.
  - eBay – valor añadido por los usuarios al dejar éstos sus valoraciones. Permite la búsqueda de elementos semejantes así como los más populares.
- Beta perpetuo: mantenimiento diario y automático. Inclusión periódica de nuevas funcionalidades y eliminación de aquéllas que los usuarios no utilizan.
- Interoperabilidad del software para permitir su uso en distintos dispositivos.

Ilustración 1 - Mapa Visual de la Web 2.0 (publicado en http://internality.com/web20/)

2.1.3 Tecnologías que apoyan la Web 2.0[2]

La evolución en las demandas de los usuarios facilitó el planteamiento de nuevos objetivos. Éstos se pudieron cumplir gracias al uso de nuevas técnicas, que apoyan a la Web 2.0. Algunas de estas técnicas que surgieron, mediante las cuales se facilita y se consiguen numerosas mejoras en la interacción de las interfaces con el usuario, son:

- CSS (Cascading Style Sheets): lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML. Estas hojas de estilo sirven de estándar para los navegadores y facilitan la separación entre la estructura y presentación de un documento. Promueven el control centralizado de un sitio Web, por lo que agilizan la actualización del mismo al tener que centrarse únicamente en el contenido.
- Técnicas de creación de aplicaciones interactivas: este tipo de aplicaciones se ejecutan en el cliente (navegador de los usuarios) mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. Esto permite que se realicen cambios sobre las páginas sin necesidad de recargarlas. Un ejemplo es AJAX (Asynchronous JavaScript and XML).
- Difusión de contenidos (RSS – RDF Site Summary): formato XML para compartir contenido en la Web. Se emplea para difundir la información recientemente actualizada o añadida a aquellos usuarios suscritos a la fuente de contenidos.
- Uso de las redes sociales para manejar usuarios y comunidades.

No debemos olvidar que la información debe poderse manipular fácilmente (agregar, modificar o eliminar contenido), y que los propios usuarios deberían poder controlar su propia información.

2.2 Estado del Arte de los Entornos Móviles

Los dispositivos móviles actuales son dispositivos multifuncionales capaces de alojar un amplio abanico de aplicaciones, tanto empresariales como personales. Las PDAs, y la

[2] Fuentes:
- http://es.wikipedia.org/wiki/Web_2.0
- http://www.maestrosdelweb.com/editorial/web2/

gran variedad de teléfonos inteligentes, además de permitir realizar llamadas telefónicas, dotan al usuario de acceso a Internet para mandar correos electrónicos, mensajes de texto, mensajería instantánea y navegación por la red; así como les ofrecen la posibilidad de crear, modificar o eliminar documentos y acceso a listas de contactos, entre otras funcionalidades.

Los dispositivos móviles se ven como una extensión a tu propio ordenador personal, ya que el trabajo realizado en ellos se puede sincronizar de manera sencilla con el ordenador para reflejar los cambios y la nueva información.

Queda fuera de este proyecto la distinción entre los distintos tipos de dispositivos móviles existentes, centrando nuestra atención únicamente en la variedad de sistemas operativos móviles.

2.2.1 Sistemas Operativos Móviles

Al igual que el sistema operativo de un ordenador, un sistema operativo móvil es la plataforma software sobre la que corren otros programas y aplicaciones. A la hora de comprar un dispositivo móvil, el fabricante habrá seleccionado el sistema operativo que haya creído conveniente para ese dispositivo específico.

El sistema operativo es el responsable de determinar las funcionalidades y características disponibles para el dispositivo, como pueden ser teclados, WAP, sincronización entre aplicaciones, correo electrónico, mensajería de texto, etc. Además, el sistema operativo determinará qué aplicaciones de terceros se podrán usar en el dispositivo.

Entre los distintos tipos de sistemas operativos móviles podemos encontrar los siguientes:

2.2.1.1 Symbian OS

Fue producto de la alianza de varias empresas de telefonía móvil (Nokia, Samsung, Sony Ericsson, etc.). Su objetivo fue crear un sistema operativo para terminales móviles que pudiera competir con el de Palm o Windows Mobile (de Microsoft).

En la actualidad, los dispositivos basados en Symbian OS acaparan el 47,2% de la cuota de mercado[3] de teléfonos inteligentes, lo que lo convierte en el sistema operativo móvil más popular.

El lenguaje nativo de Symbian es C++, cuyo aprendizaje puede ser complejo debido a que este sistema operativo requiere del uso de técnicas específicas como descriptores o la pila de limpieza (cleanup stack[4]). Sin embargo, soporta numerosos lenguajes de programación, por lo que desarrollar aplicaciones para dispositivos Symbian puede convertirse en una tarea sencilla:

- C++
- Java
- Flash Lite
- Python, Ruby
- OPL
- PIPS

Diseño:

Symbian implementa multitarea preventiva (es el propio procesador quien asigna los tiempos de CPU a las tareas en ejecución) y protección de memoria.

Este sistema operativo se creó teniendo en mente los siguientes tres principios de diseño:

- La integridad y la seguridad de los datos de usuario es de vital importancia.
- No se debe perder el tiempo del usuario.
- Todos los recursos son pocos.

[3] Estadísticas de Abril de 2010 que reflejan las ventas a lo largo del año 2009.
[4] Funcionalidad de Symbian C++ para manejar la destrucción de objetos y recursos en caso de condiciones de excepción.

Para seguir estos principios, se emplea un microkernel (en el que sólo mínimas partes del sistema corren en modo núcleo), una aproximación de retrollamada[5] a servicios, y se mantiene la separación entre la interfaz de usuario y el motor.

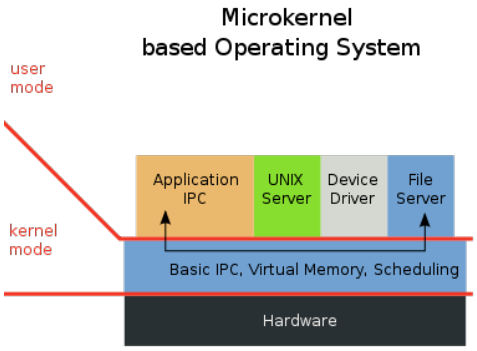


Ilustración 2 - Estructura de un Sistema Operativo basado en microkernel

Este sistema operativo está optimizado para dispositivos con baterías de baja potencia y sistemas ROM. Tanto el sistema operativo como las aplicaciones siguen un diseño orientado a objetos: Modelo-Vista-Controlador (MVC).

Hace especial hincapié en la conservación de recursos. Asimismo, existen técnicas de conservación de espacio de disco, a pesar de que los discos de los dispositivos Symbian son, normalmente, memorias flash.

Cabe destacar que la programación Symbian se basa en eventos y que el procesador conmuta a un modo de bajo consumo cuando las aplicaciones no están tratando directamente con un evento. Esto último, gracias a los "objetos activos" [6], es una forma de multitarea cooperativa.

El núcleo de Symbian (EKA2[7]) soporta una respuesta en tiempo real lo suficientemente rápida como para construir un teléfono de un solo procesador de núcleo único a su alrededor. Esto ha permitido el desarrollo de teléfonos Symbian más pequeños, baratos y eficientes que sus predecesores.

---

[5] El propósito del patrón de retrollamada es permitir que un cliente se registre en un servidor para ciertas operaciones. De esta forma, recibirá una notificación por parte del servidor cuando finalice la operación.

[6] Los objetos activos pueden realizar peticiones de servicios asíncronos. Cuando se lleva a cabo esta petición, se devuelve el control al objeto llamante (a través de la referencia que debe haber incluido el llamante a sí mismo previamente).

[7] EPOC Kernel Architecture 2: segunda generación del núcleo de la plataforma Symbian.

Arquitectura:

La arquitectura en la que se basa Symbian consta de cinco capas:

Ilustración 3 - Arquitectura de Symbian

Entre los servicios del sistema operativo se encuentran los servicios genéricos, los de comunicaciones, gráficos y multimedia, y los de conectividad.

La capa de servicios base es el nivel más bajo al que tienen acceso las operaciones de usuario. Incluye un servidor de ficheros y las librerías de usuario, un gestor de plugins, un repositorio central, gestor de base de datos y servicios criptográficos.

La arquitectura de microkernel asegura que sólo el mínimo necesario se mantiene en el núcleo para maximizar la robustez, disponibilidad y respuesta. Contiene un planificador, gestor de memoria y drivers de dispositivos, pero el resto de servicios (como conectividad, telefonía y soporte de sistema de ficheros) se encuentran en las capas de servicios del sistema operativo y servicios básicos.

Modelo de Producción:

Se proporcionan SDKs (kits de desarrollo de software) que contienen documentación, los ficheros de cabecera y librerías necesarios para desarrollar software para Symbian,

y un emulador de ventanas (WINS). Además, para la programación en Symbian C++, se emplean, comúnmente, entornos de desarrollo integrados.

Una vez desarrolladas, las aplicaciones para Symbian deben "encontrar" un camino hacia los terminales móviles clientes. Se empaquetan en ficheros SIS que pueden instalarse por medio de una conexión con el ordenador, Bluetooth o tarjeta de memoria.

Existe un programa para firmar aplicaciones que dispongan de funcionalidades complejas, obligatorio para las versiones 9.x y que aseguran una cierta fiabilidad: Symbian Signed. Para poder usar algunas de las funcionalidades más llamativas en sus aplicaciones, los desarrolladores deben pagar una cuota, lo que, comparándolo con plataformas como Palm OS, Windows Mobile y Android, está convirtiéndola en una plataforma poco popular para proyectos Open Source y desarrolladores independientes.

2.2.1.2 Windows Phone

Antes conocido como Windows Mobile, este sistema operativo ha sufrido, recientemente, un cambio de nombre aprovechando el lanzamiento de Windows Mobile 6.5 y con la previsión del lanzamiento de Windows Phone 7 (sistema operativo que se tratará para la comparación en este proyecto).

Se trata de un sistema operativo móvil, desarrollado por Microsoft, para uso en teléfonos inteligentes y dispositivos móviles. Su versión actual está basada en el núcleo del sistema operativo Windows CE 7 e implementa una serie de aplicaciones básicas desarrolladas con la API de Microsoft Windows.

Siguiendo en la trayectoria de Microsoft, estas aplicaciones están diseñadas para mantener la estética y el funcionamiento de las aplicaciones de las versiones de escritorio de Windows. Adicionalmente, existe software desarrollado por terceros compatible con Windows Phone y se ha desarrollado el "Wndows Phone Marketplace", para facilitar la compra de dicho software (similar a lo que ocurriera con el Apple Store, de iPhone, y el Market, de Android).

La cuota de mercado referente a los dispositivos basados en Windows Mobile ha ido decrementando año tras año, llegando, en la actualidad, a acaparar alrededor del 8% del mercado mundial.

El lenguaje de programación soportado es C++.

Arquitectura:

Hasta ahora, toda la información que se tiene sobre la arquitectura del sistema operativo móvil de Microsoft es la que se ha filtrado y ha llegado a la Web.

En los documentos filtrados se hace constar que Windows Phone 7 es un sistema operativo de 32-bit con una arquitectura de doble capa formada por una capa del núcleo y una capa de usuario.

A los procesos de las aplicaciones se les dará hasta 1GB de memoria virtual con un total de 2GB de memoria alojada a los procesos, mientras que al núcleo se le conceden otros 2GB.

Modelo de Producción:

Windows Phone permite el desarrollo de software de terceros. Los desarrolladores disponen de varias alternativas para desplegar una aplicación: código en Visual C++, código gestionado[8] compatible con el ".NET Compact Framework" [9], o código servidor que pueda ser desplegado empleando el Internet Explorer Móvil, o cualquier otro cliente móvil en el dispositivo del usuario.

Habitualmente, Microsoft lanza SDKs que funcionan conjuntamente con su entorno de desarrollo Visual Studio. Estos kits de desarrollo incluyen imágenes de emuladores sobre las que los desarrolladores pueden probar y depurar sus aplicaciones. De cara a

[8] Código de un programa de ordenador que se ejecuta bajo la gestión de una máquina virtual. El código no gestionado es aquél ejecutado directamente por la CPU del ordenador.
[9] Versión del .NET Framework diseñado para ejecutar en dispositivos móviles basados en Windows CE.

la nueva versión del sistema operativo de Microsoft, la SDK de aplicaciones para Windows Phone 7 se basará en "Silverlight" [10], "XNA" [11] y .NET Compact Framework. Las herramientas que se usarán principalmente para el desarrollo serán el entorno de desarrollo Visual Studio 2010 y "Expression Blend" [12].

Tal y como se ha mencionado anteriormente, Microsoft ha abierto un servicio de distribución de aplicaciones de terceros (Windows Phone Marketplace), que permite a los usuarios buscar y descargar este tipo de aplicaciones.

Los desarrolladores podrán obtener el 70% del beneficio que proporcionen sus aplicaciones o incluir un modelo de publicidad en sus aplicaciones. Por tanto, podrán presentarse aplicaciones gratuitas, de pago, o gratuitas con publicidad.

Para publicar las aplicaciones creadas por los desarrolladores, éstos deberán contar con una cuenta en el sitio oficial de aplicaciones para Windows Phone 7, lo que les supondrá un coste anual de 99$. Por otro lado, el envío de aplicaciones por parte de estudiantes es gratuito: a través del programa "DreamSpark" de Microsoft, en el que deberán verificar su identidad.

Además, todas las aplicaciones seguirán un proceso de verificación a través del cual Microsoft se asegurará de que no se presenten aplicaciones con contenidos dañinos moralmente.

2.2.1.3 iPhone OS (iOS)

Sistema operativo móvil desarrollado y comercializado por Apple Inc. Es el sistema operativo por defecto (y el único soportado) del iPhone, iPod Touch y del iPad.

[10] Framework de aplicaciones Web que proporciona funcionalidades que integran multimedia, gráficos, animaciones e interacción en un único entorno de ejecución. (Similar a las funcionalidades de Adobe Flash).
[11] Conjunto de herramientas que facilita el desarrollo y gestión de juegos.
[12] Herramienta de diseño de interfaces de usuario para la creación de interfaces gráficas para aplicaciones Web y de escritorio.

Derivado del Mac OS X[13], con el que comparte la base de Darwin OS[14], es un sistema operativo basado en Unix por naturaleza.

En el último año, la cuota de mercado mundial obtenida por el sistema operativo de Apple ha estado rozando el 15%, convirtiéndolo en el tercer sistema operativo preferido por los usuarios, tras Symbian y BlackBerry.

Arquitectura:[15]

La arquitectura del iOS es similar a la arquitectura básica de Mac OS X. A alto nivel, el iOS actúa como intermediario entre el hardware y las aplicaciones de pantalla.

A más bajo nivel, la implementación de las tecnologías de iPhone OS se puede ver como un conjunto de cuatro capas de abstracción: la capa del núcleo del sistema operativo, la capa de servicios centrales, la capa de medios y la capa "Cocoa Touch" [16].

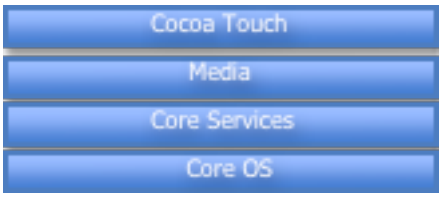


Ilustración 4 - Arquitectura iPhone OS

Capa Cocoa Touch:

Es una de las capas más importantes de iPhone OS. Incluye los frameworks clave que proporcionan la infraestructura necesaria para desarrollar aplicaciones en iPhone OS, de tal manera que éstas conserven la apariencia y comportamiento del propio sistema operativo.

[13] Conjunto de sistemas operativos e interfaces de usuario basados en Unix y desarrollados y comercializados por Apple Inc.
[14] Sistema operativo de código abierto lanzado por Apple Inc. en el año 2000. Forma los componentes centrales sobre los que se basan Mac OS X, Apple TV e iOS.
[15] Fuentes:
http://developer.apple.com/iphone/library/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/IPhoneOSOverview/IPhoneOSOverview.html
[16] API para construir programas software que ejecutan sobre el iPhone, iPod Touch y el iPad.

Cocoa emplea el patrón de diseño Modelo-Vista-Controlador. Los modelos encapsulan los datos de aplicaciones, las vistas muestran y editan los datos, y los controladores median la lógica entre los dos.

Emplea el lenguaje de programación Objective-C, un lenguaje de programación orientado a objetos compilado para ejecutar a grandes velocidades, y superconjunto de C.

Los elementos más destacados que se pueden conseguir a través de estos frameworks son:

- Soporte para manejar eventos y controles "multi-touch"
- Uso e integración del acelerómetro
- Web view – vistas Web embebidas en las aplicaciones

Capa de Medios:

En esta capa se encuentran las tecnologías de gráficos, audio y vídeo pensadas para poder crear una buena experiencia multimedia en el dispositivo móvil.

Capa de Servicios Centrales:

Proporciona los servicios de sistema fundamentales que emplean todas las aplicaciones. Gran parte del sistema se construye sobre ellos.

Modelo de Producción:

Apple Inc. dispone de un kit de desarrollo que permite a los desarrolladores crear aplicaciones para este sistema operativo. El entorno de desarrollo para la SDK de iPhone es Xcode 3.1, en el que el lenguaje de desarrollo es Objective-C. Asimismo, incluye un emulador de iPhone para poder probar dichas aplicaciones.

Sin embargo, sólo se puede instalar una aplicación en un dispositivo real tras haber pagado una cuota de desarrollador de aplicaciones iPhone.

Similarmente a lo que sucederá con Windows Phone 7, los desarrolladores pueden fijar un precio, por encima de un mínimo establecido, para sus aplicaciones, que se distribuirán a través del App Store. De este precio, los desarrolladores recibirán un 70% de los beneficios. De forma alternativa, un desarrollador puede decidir distribuir su aplicación de manera gratuita, en cuyo caso sólo deberán pagar la cuota de miembro, y no los costes de publicación y distribución.

2.2.1.4 BlackBerry OS

Se trata de una plataforma de software propietario, creado por RIM[17] para su línea de teléfonos inteligentes BlackBerry. Este sistema operativo proporciona multitarea y soporta dispositivos de entrada especializados (adoptados por RIM en sus dispositivos móviles), como pueden ser la rueda, bola y panel de desplazamiento, o la pantalla táctil.

En cuanto a la cuota de mercado de BlackBerry OS, gracias a su capacidad de sincronización con los servidores de correo corporativos, con Microsoft Exchange, Lotus Domino, calendario, tareas, notas y contactos, acapara aproximadamente el 20% de los teléfonos inteligentes vendidos en el mundo.

Mientras que algunos dispositivos de BlackBerry están programados en lenguaje C++, los nuevos que están surgiendo soportan la plataforma Java 2 Micro Edition (J2ME), ya que la tecnología Java hace de la programación de aplicaciones una tarea más sencilla: los desarrolladores pueden centrarse más en la lógica dejando de lado la preocupación por la gestión de memoria, y proporciona una independencia en la plataforma clave.

[17] Research In Motion – Compañía de telecomunicaciones y dispositivos inalámbricos canadiense, más conocida por desarrollar el teléfono inteligente BlackBerry.

Arquitectura:

La arquitectura del sistema operativo de BlackBerry es, y siempre ha sido, un misterio. Existe una gran preocupación por mantenerla en secreto. De ahí que no existan virus ni otros malware capaces de instalarse en sus dispositivos.

Por esta razón, lo único que se puede comentar sobre su arquitectura es que dispone del software "BlackBerry Enterprise Server" (BES), encargado de llevar a cabo la sincronización inalámbrica con el calendario, tareas, contactos, correo electrónico y notas.

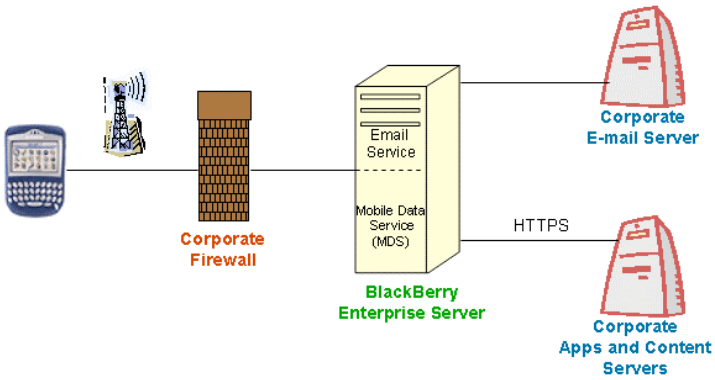


Ilustración 5 - Arquitectura BlackBerry

Una alternativa al servidor BES es el "BlackBerry Internet Service" (BIS), que proporciona acceso a Internet al usuario.

Modelo de Producción:

BlackBerry OS dispone de una API, a través de cuyas clases los desarrolladores pueden escribir código fuente para sus aplicaciones.

Para proporcionarles flexibilidad en el diseño de aplicaciones empresariales inalámbricas sofisticadas, BlackBerry soporta dos modelos de aplicaciones:

- El modelo basado en navegador, que permite a los desarrolladores centrarse en desarrollar contenido en un lenguaje de marcado estándar. Esto le libera de

preocuparse por la interfaz cliente, pero limita la funcionalidad a aquello que soporte el navegador.

- Aplicaciones Java personalizadas, que permite al desarrollador crear interfaces de usuario personalizadas y soporta contenido más allá de texto e imágenes. También se pueden desarrollar aplicaciones que un usuario puede descargar e instalar en dispositivos inalámbricos. Además, algunos dispositivos BlackBerry vienen con un conjunto de interfaces de programación de aplicaciones (API) y herramientas que permiten crear aplicaciones Java personalizadas.

El entorno de desarrollo de Java para BlackBerry (JDE) es un entorno de desarrollo integrado (IDE) que proporciona las herramientas necesarias para desarrollar aplicaciones Java que ejecuten sobre dispositivos BlackBerry.

El JDE necesita el conjunto de desarrollo software Java 2 para poder ejecutar, y viene con un simulador BlackBerry para pruebas.

A través del JDE se puede compilar el código fuente, empaquetarlo en un fichero con extensión ".cod" (formato propietario), y cargar la aplicación en el dispositivo, cuya máquina virtual Java ejecutará el archivo. Se debe tener en cuenta que, al igual que en otros entornos, se produce un proceso de preverificación antes de cargar las clases en el terminal. Asimismo, el JDE preverifica el código automáticamente antes de empaquetarlo en los ficheros ".cod".

2.2.1.5 Google Android

Sistema operativo móvil creado por la Open Handset Alliance, que lidera Google, y desarrollado sobre un núcleo Linux. Tiene licencia software Apache de software libre y código fuente abierto.

Este sistema operativo es una pila de software que incluye un sistema operativo, una serie de aplicaciones esenciales y middleware.

Android permite a los desarrolladores escribir código manejado en lenguaje de programación Java, que controla el dispositivo a través de librerías Java desarrolladas

por Google. Se trata, además, de una plataforma abierta que permite a cualquier fabricante desarrollar sus productos sobre ella.

Su pila de software consiste de aplicaciones Java que ejecutan en un framework de aplicaciones Java orientadas a objetos sobre librerías de núcleo Java que ejecutan sobre una máquina virtual Dalvik[18] con compilación JIT[19].

A pesar de sus grandes competidores (BlackBerry OS y iPhone OS), Android ha sabido abrirse un hueco en el mundo de los teléfonos inteligentes. Hasta finales del 2009, las ventas mundiales de dispositivos con este sistema operativo ascendían al, aproximadamente, 5% del mercado. Sin embargo, recientes estudios de importantes empresas de estudio de mercado han desvelado un cambio en la tendencia de los teléfonos inteligentes.

Sólo en Estados Unidos, la información del primer cuatrimestre de 2010 desarrollada por el grupo NPD, ha revelado que Android OS ocupa el segundo lugar de ventas (28%), tras BlackBerry RIM (36%) y superando al sistema operativo de Apple (21%)[20]. Por otro lado, Gartner Inc. predijo, en octubre de 2009, que Android se convertiría en la segunda plataforma más popular del mundo, detrás de Symbian OS.

Arquitectura[21]:

A continuación se muestran los componentes principales del sistema operativo Android:

[18] MV que ejecuta aplicaciones convertidas a un formato ejecutable compacto Dalvik (.dex) apto para sistemas cuyas capacidades de memoria y velocidad del procesador están restringidas.
[19] Just-in-time compilation: técnica para mejorar el rendimiento de un programa informático. Aproximación híbrida, entre compilación interpretada y estática, en la que la traducción tiene lugar continuamente (interpretada) pero con el cacheo de código traducido para minimizar la degradación del rendimiento.
[20] http://www.npd.com/press/releases/press_100510.html
[21] Fuente: http://developer.android.com/intl/fr/guide/basics/what-is-android.html

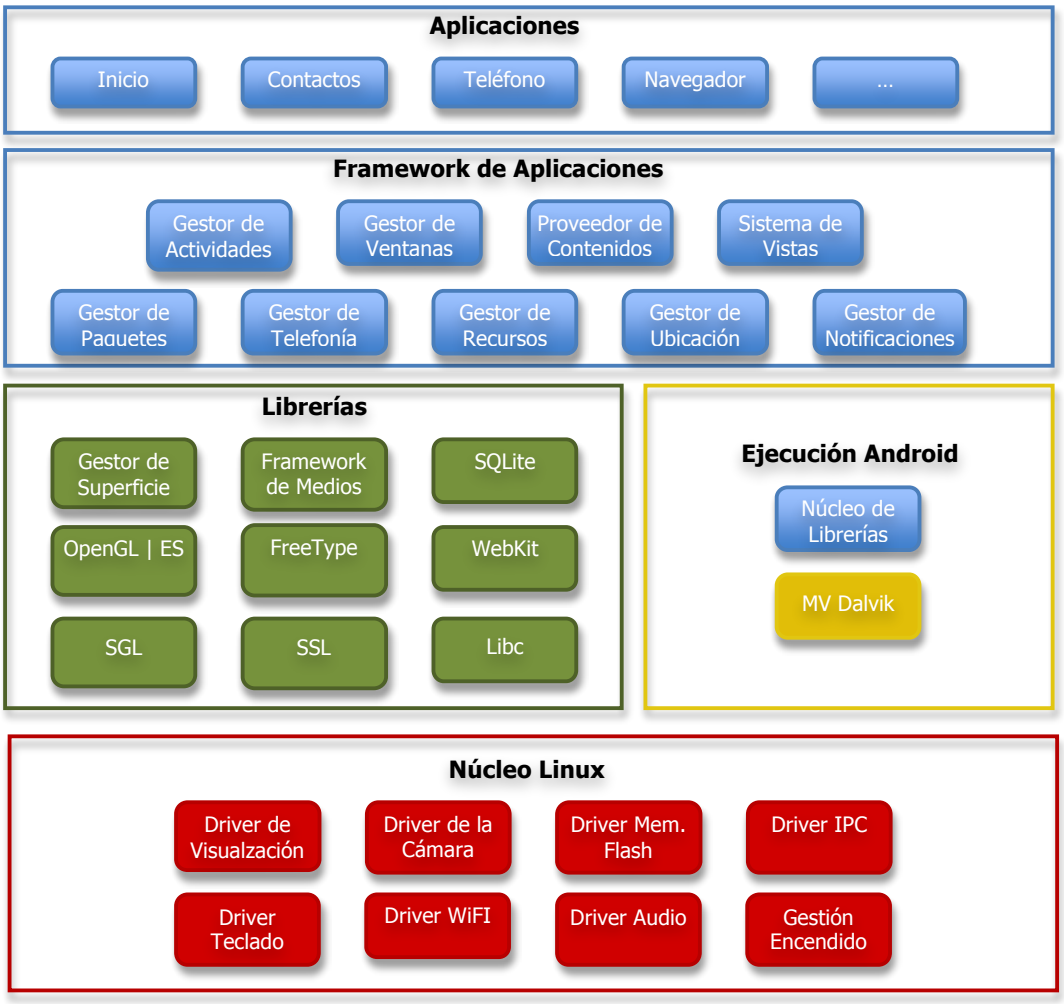


Ilustración 6 - Arquitectura SO Android

Aplicaciones:

Android posee un conjunto de aplicaciones esenciales entre las que podemos encontrar un cliente de correo electrónico, programa de mensajería, calendario, mapas, navegador de Internet y contactos.

Todas estas aplicaciones están escritas en lenguaje de programación Java.

Framework de aplicaciones:

Al tratarse de una plataforma de desarrollo abierta, Android ofrece a los desarrolladores la posibilidad de construir aplicaciones extremadamente innovadoras y ricas. Los desarrolladores pueden aprovecharse del hardware del dispositivo, acceder a información geoposicional, ejecutar servicios en segundo plano, añadir notificaciones a la barra de estado, etc.

Asimismo, el framework de APIs al que tienen acceso los desarrolladores es el mismo que el utilizado en las aplicaciones básicas. La arquitectura de aplicaciones está diseñada de tal forma que simplifica la reutilización de componentes: cualquier aplicación puede publicar sus capacidades y cualquier otra aplicación podrá usarlas (siempre dentro de unas restricciones de seguridad forzadas por el framework).

Bajo todas las aplicaciones se alojan una serie de servicios y sistemas, entre los que se encuentran:

- Vistas: usadas para la construcción de una aplicación, ya que incluye listas, tablas, cajas de texto, botones e incluso un navegador embebido en la aplicación.
- Proveedor de contenidos: permite a las aplicaciones acceder a los datos de otras aplicaciones (como, por ejemplo, los contactos) o compartir sus propios datos.
- Gestor de recursos: proporciona acceso a recursos como cadenas de texto, gráficos y ficheros de maquetación de la aplicación.
- Gestor de notificaciones: permite a las aplicaciones mostrar alertas en la barra de notificaciones.
- Gestor de actividades: gestiona el ciclo de vida de las aplicaciones y proporciona una pila común de navegación.

Librerías:

Android incluye un conjunto de librerías C/C++ utilizadas por varios componentes del sistema Android. Sus funcionalidades se exponen a los desarrolladores a través del framework de aplicaciones.

Tiempo de Ejecución:

Android incluye un conjunto de librerías esenciales que proporcionan la mayor parte de la funcionalidad disponible en las librerías básicas de Java.

Toda aplicación ejecuta en su propio proceso, con su propia instancia de la máquina virtual Dalvik, que permite que un dispositivo pueda ejecutar múltiples máquinas virtuales a la vez eficientemente.

Núcleo Linux:

El sistema operativo Android se basa en la versión 2.6 de Linux para sus servicios centrales tales como seguridad, gestión de memoria, gestión de procesos, pila de conexión y modelo de drivers. Además, el núcleo actúa como una capa de abstracción entre el hardware y el resto de la pila de software.

Modelo de Producción:

Android incluye una SDK con una serie de herramientas de desarrollo: depurador, librerías, emulador de dispositivo móvil, documentación, código de ejemplo y tutoriales. Para poder desarrollar una aplicación para Android se debe disponer de lo siguiente:

- Ordenador con arquitectura x86 (Linux; Mac OS X 10.4.8, o superior; Windows XP o Vista)
- JDK
- Apache Ant
- Python 2.2 o posterior

El entorno de desarrollo integrado oficial es Eclipse (3.2 o superior) usando el plugin de las herramientas de desarrollo de Android (ADT), aunque se permite que los desarrolladores empleen cualquier editor de textos para editar ficheros Java y XML y utilizar, posteriormente, las herramientas de línea de comando.

Las aplicaciones Android se empaquetan en formato .apk y se almacenan en /data/app en el sistema operativo.

De cara a poder distribuir las aplicaciones de terceros, Android permite la publicación de las mismas en su "Android Market", en el que un desarrollador debe subir sus aplicaciones firmadas (para verificar su origen). Éstas aplicaciones pueden ser de pago, gratuitas o con publicidad integrada. Eso sí, se debe pagar una cuota de 25$ para obtener una cuenta de desarrollador.

Una vez publicada una aplicación, ésta será visible en el Market y se le proporcionarán estadísticas al desarrollador en cuanto a valoración, número de descargas, etc.

2.2.2 Tabla Comparativa de SSOO Móviles

A continuación se muestra una tabla comparativa de los sistemas operativos móviles expuestos en la sección anterior.

	Symbian OS	**Windows Phone**	**iPhone OS**	**BlackBerry OS**	**Android OS**
Cuota de mercado mundial	47,2%	8%	15%	20%	5% (tendencia a subir)
Lenguaje de programación SO	C++	C++	Objective-C	C++ (últimamente Java)	Java / XML
Lenguaje de programación aplicaciones	Múltiples	C++	Objective-C	Java	Java / XML
Ofrece SDK	Sí	Sí	Sí	Sí	Sí
Ofrece emulador	Sí	Sí	Sí	Sí	Sí
Núcleo	Symbian (EKA2)	Windows CE	XNU	Propietario	Linux
Multitarea	Sí (preventiva)	Sí	Sí (últimas versiones)	Sí	Sí
Aplicaciones en segundo plano	Sí	Sí	Sí	Sí	Sí
Notificaciones en segundo plano	No	Sí	No	Sí	Sí

	Symbian OS	Windows Phone	iPhone OS	BlackBerry OS	Android OS
"Tienda" de aplicaciones	No	Sí	Sí	Sí	Sí
Código abierto	Sí	No	No	No	Sí
Multitáctil	No	No	Sí	Sí	Sí (no oficial)

Tabla 1 - Comparativa de los sistemas operativos móviles más populares

2.3 Aplicación GeoBuddies Existente

Este proyecto no surge de la nada, sino que se basa en una idea ya existente: GeoBuddies. GeoBuddies representa el proyecto desarrollado para plataformas Symbian por la Universidad de Santiago de Compostela en colaboración con la Universidad Politécnica de Madrid, proyecto financiado por el Ministerio de Educación y Ciencia (MEC).

La idea tras este proyecto era la del desarrollo de una infraestructura tecnológica que permitiese a los peregrinos del Camino de Santiago acceder a información en tiempo real (siempre que las infraestructuras de comunicaciones lo permitan) sobre lugares y servicios de interés a lo largo de la ruta. La manera de conseguir esto era desarrollando una aplicación **móvil** con las funcionalidades de comunicación y gestión de recursos necesarias.

Para permitir la compartición de experiencias entre los distintos peregrinos, se creó una comunidad virtual de usuarios del Camino de Santiago en la que, sus miembros, a través de sus dispositivos móviles, pudieran introducir y consultar esta información. Se debe tener en cuenta que, entre toda esta información, cobra un papel importante la información relativa a la **posición geográfica** y las **anotaciones semánticas** de los recursos que permita almacenarla en la comunidad virtual.

Además de la comunidad virtual, existe una arquitectura orientada a servicios (SOA) que permite, a través de los servicios Web, a los usuarios acceder a toda la información disponible existente del lugar en el que se encuentren en cualquier momento.

2.3.1 Arquitectura de la Aplicación

La arquitectura existente está basada en dos niveles: una capa cliente o interfaz de acceso, y una capa de servicios que proporciona la información del sistema. Su fin es el de satisfacer los requisitos de desarrollo, interfaz y nivel de servicio del sistema:

- Permitir que el sistema proporcione información remota a los dispositivos móviles.
- Permitir que mantenga su funcionalidad aún en ausencia de cobertura de red.
- Reducir los costes de conexión del terminal.
- Limitar el conjunto de recursos software necesarios para la ejecución de la aplicación.

2.3.2 Servicios Proporcionados

La aplicación existente ofrece una funcionalidad determinada basándose en las necesidades de los peregrinos del Camino de Santiago. Así, se pretende ofrecer una mayor diversificación de las rutas disponibles (atendiendo a los gustos y experiencias de otros peregrinos en nuestra comunidad de usuarios), se ofrece la posibilidad de encontrar lugares y servicios de interés así como proveer al usuario de información en tiempo real.

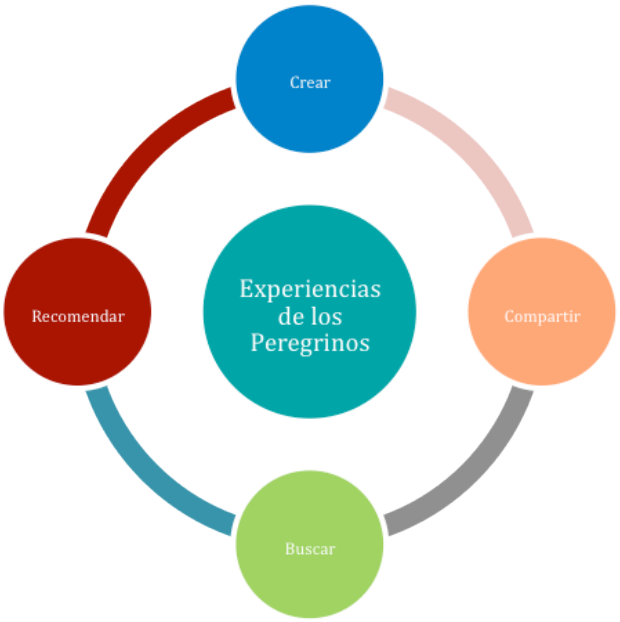


Ilustración 7 - Funcionalidad básica

Entre los requisitos funcionales básicos se encuentran: crear una cuenta de usuario en la comunidad virtual, iniciar sesión, introducir recursos base (rutas, imágenes, vídeos, POIs, etc.), definir un conjunto de peregrinos amigos, valorar recursos o buscar recursos según los siguientes criterios

de búsqueda:

- Tipo de recurso. Una vez establecido, se puede realizar la búsqueda o indicar el siguiente criterio.
- Término de búsqueda. Una vez establecido, se puede realizar la búsqueda o indicar el siguiente criterio.
- Localización. Tras informarlo, se procede con la búsqueda. Si la localización es indispensable para obtener el recurso, se obliga al usuario a rellenar este campo.

A continuación se muestran algunas de las capturas de pantalla de la aplicación durante su ejecución en un dispositivo real.

2.3.2.1 Pantalla Principal

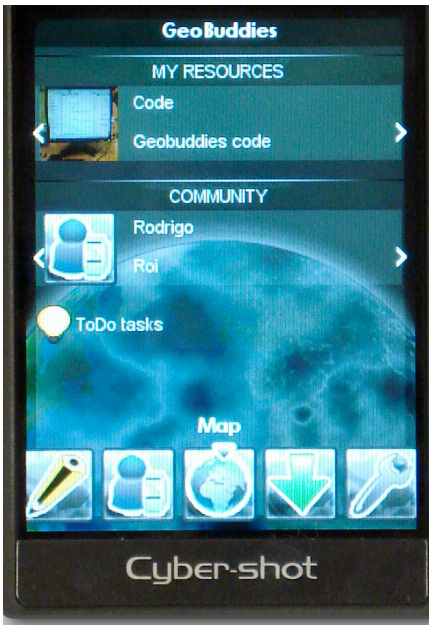


Ilustración 8 - Pantalla principal

A la izquierda se puede observar la pantalla principal de la aplicación GeoBuddies existente. Se puede apreciar una barra de menú deslizante en la parte inferior de la pantalla, así como dos secciones de acceso rápido a la "Comunidad" y a los "Recursos" del usuario.

Asimismo, como parte de la funcionalidad, existe la posibilidad de establecer la sincronización tardía de recursos con la comunidad virtual. Esto es de vital importancia por la existencia de posibles zonas sin red móvil a lo largo del Camino de Santiago.

Por ello, se distinguirá un indicador en forma de bombilla encendida cuando el usuario haya agregado nuevos recursos que no se hayan sincronizado.

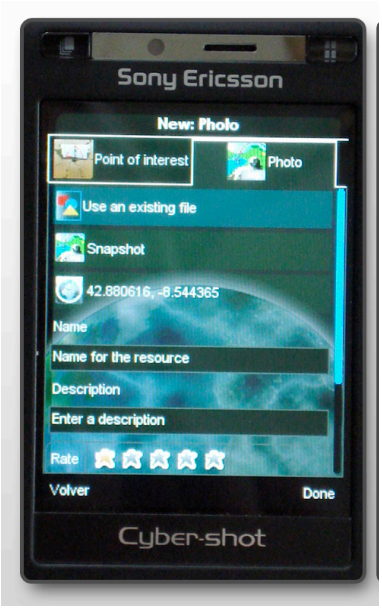
2.3.2.2 Vista de Mapa

A continuación se muestra la vista de mapa, en la que se puede apreciar un recurso sobre él y la información que lo detalla en la parte inferior de la pantalla.

Como se puede leer en el segundo globo de información, al emplear el joystick del terminal en una determinada dirección, el usuario podrá pasar de un elemento a otro del mapa.

Ilustración 9 - Vista de mapa

2.3.2.3 Pantalla de Nuevo Recurso



En la siguiente imagen se puede distinguir el formulario de alta de un nuevo recurso, más concretamente una fotografía, en la aplicación.

El usuario deberá elegir la ruta de la imagen, o bien tomar una nueva foto, así como informar todos aquellos campos que considere necesarios.

Ilustración 10 - Nueva fotografía

3. Objetivos del Proyecto

Al igual que ocurriera con la aplicación existente, el objetivo de la aplicación que se ha desarrollado para este proyecto no es otro que el de proveer a los peregrinos del Camino de Santiago de la información relativa a los lugares por los que pase el trazado de su ruta:

- Trazado y perfil de cada etapa
- Lugares de interés cultural
- Hospedaje
- Restauración
- Otros servicios (atención médica, ocio, etc.)

La mejor manera de conseguir distribuir esta información es creando una comunidad de peregrinos, unidos por un mismo objetivo, que compartan sus experiencias en el camino.

Por ello, se debe migrar la idea de la creación de una comunidad virtual de peregrinos así como la arquitectura orientada a servicios.

Al poseer ya toda una colección de servicios Web que permiten a los usuarios de la aplicación informar y consultar información de su interés, este proyecto se ha centrado en la definición de la arquitectura necesaria para el desarrollo de una aplicación, similar a la existente, en plataformas Android, con el fin de expandir el uso de la aplicación a cuantos más usuarios mejor.

Asimismo, ha sido necesario identificar la mejor opción soportada por esta plataforma para la comunicación entre el dispositivo móvil y los servicios Web existentes, e integrar la interfaz de usuario con todas las posibilidades que ofrece el sistema operativo Android.

Se trata, por tanto, de construir un sistema de compartición de recursos en el que los usuarios podrán dar valoraciones y recomendaciones generales para dejar constancia

de sus experiencias en una comunidad virtual. Para su desarrollo, se tendrán en cuenta los siguientes aspectos:

- Se debe presentar la posibilidad de manejar la aplicación en modo sin conexión, ya que en zonas rurales por las que pueda discurrir el Camino de Santiago podría no haber cobertura (ni 3G ni GPRS).
- Las búsquedas que realice un usuario serán, necesariamente, geográficas. Es decir, se presentarán aquellos recursos que cumplan el filtro de la búsqueda y que se encuentren cerca de la ubicación del usuario.
- Sistema de alertas que recuerde al usuario que tiene recursos pendientes de registrar en la comunidad virtual (en caso de que la aplicación haya estado funcionando en modo offline).
- Se facilitará el acceso a los contactos favoritos (amigos) y a los recursos personales del usuario.
- Se ofrecerá al usuario la posibilidad de buscar recursos sobre un mapa.

3.1 Objetivos Generales

- Iniciar la aplicación sin necesidad de estar conectado (siempre y cuando el usuario se haya registrado previamente en la aplicación).
- Gestionar el alta de usuarios nuevos en la comunidad virtual.
- Gestionar el alta de nuevos recursos (tanto a través del menú como sobre el mapa).
- La aplicación permitirá a los usuarios gestionar sus peregrinos "amigos".
- Se ofrecerá la posibilidad de valorar los diferentes recursos.

3.2 Objetivos Específicos

- Comunicar la interfaz de usuario con los servicios Web disponibles.
- Definir los posibles escenarios y casos de uso.
- Definir las incidencias que se puedan producir.
- Realizar el prototipado de la interfaz de usuario.

- Gestionar la comunicación entre la base de datos interna del dispositivo (SQLite) y la interfaz de usuario.
- Conocer en todo momento el estado de los recursos del usuario (si se han publicado o no en la comunidad).
- La aplicación deberá ser capaz de mostrar los recursos sobre el mapa, así como de permitir que éstos sean navegables.

4. Especificación de Requisitos Software

Este capítulo pretende poner de manifiesto los requisitos de la aplicación GeoBuddies para Android.

El objetivo de la ERS (Especificación de Requisitos Software) es definir de una manera clara y formal todas las funcionalidades y restricciones del sistema que se desea construir.

4.1 Requisitos de Arquitectura

R1. El sistema debe correr sobre terminales móviles con sistema operativo Android 1.5 o superior.

R2. El sistema debe contener una base de datos SQLite para el almacenamiento local de recursos.

4.2 Requisitos Funcionales

4.2.1 Requisitos de Usuario

R3. El usuario podrá iniciar la aplicación sin necesidad de conexión.

R4. El usuario podrá dar de alta nuevos recursos sin necesidad de conexión.

R5. El usuario podrá consultar sus recursos sin necesidad de conexión.

R6. El usuario podrá dar de alta nuevos recursos a través del menú de opciones: rutas, puntos de interés, fotografías, valoraciones, etc.

R7. El usuario podrá dar de alta nuevos recursos sobre el mapa.

R8. El usuario podrá introducir fotografías que luego utilizará como recurso.

R9. El usuario podrá definir un conjunto de amigos.

R10. El usuario podrá anotar recursos (POIs, fotos, eventos y rutas) a través de etiquetas y comentarios.

R11. El usuario podrá valorar recursos.

R12. El usuario podrá realizar una búsqueda de recursos a través del menú de opciones.

R13. El usuario podrá realizar una búsqueda de otros peregrinos.

R14. El usuario podrá elegir si enviar un recurso a la red o almacenarlo únicamente en el dispositivo.
R15. El usuario podrá consultar los recursos de los usuarios de su comunidad (amigos).
R16. El usuario podrá visualizar las rutas de los usuarios sobre un mapa navegable y con funciones de zoom.
R17. El usuario podrá registrarse en la aplicación a través de la pantalla de login.
R18. El usuario deberá identificarse antes de la utilización del sistema.
R19. El usuario podrá elegir si desea que el sistema recuerde los datos de inicio de sesión.
R20. El usuario podrá acceder a una vista de mapa y navegar por él para visualizar recursos de usuarios cercanos geográficamente.
R21. El usuario conectado podrá modificar sus preferencias de inicio de sesión.

4.2.2 Requisitos de Sistema

R22. El sistema debe ser capaz de generar pantallas con los resultados de las consultas del usuario.
R23. El sistema deberá presentar menús de opciones para las distintas pantallas.
R24. El sistema mostrará un menú de opciones fijo en la pantalla principal.
R25. Las listas de resultados deberán ser navegables con el fin de adaptar los contenidos al tamaño de la pantalla.
R26. La aplicación será capaz de mostrar mapas geoanotados.
R27. La aplicación será capaz de acceder a los datos almacenados en el terminal para obtener rutas que el usuario utilizará como recursos.
R28. El sistema deberá tener acceso a la red.
R29. El sistema deberá tener acceso a los mapas de Google.
R30. La aplicación deberá tener acceso a los servicios de geolocalización (GPS) y detectará si están o no activos. Se informará al usuario mediante un mensaje.
R31. El sistema deberá tener acceso al software de la cámara del terminal.
R32. El sistema será capaz de detectar si tiene cobertura e informará al usuario mediante un mensaje.

Los requisitos R27, R28, R29, R30 y R31 hacen referencia al funcionamiento específico de la plataforma Android, en la que, para utilizar determinados recursos, se deben establecer los permisos necesarios en el manifiesto de la aplicación. Todos los permisos que aquí se detallen, se solicitarán al usuario final a la hora de instalar la aplicación.

4.2.3 Requisitos de Interfaz

R33. Interfaz accesible e intuitiva

R34. Interfaz consistente con la aplicación GeoBuddies para Symbian.

R35. Internacionalización de la interfaz (según la configuración del terminal, se ofrecerá en español y en inglés).

R36. El diseño de la interfaz debe reconocer eventos de pantalla y teclado para la navegación del sistema.

R37. La comunicación con los servicios Web se realizará por medio del protocolo simple de acceso a objetos (SOAP). Se deberá implementar una versión ligera para dispositivos móviles o buscar unas librerías ya adaptadas.

4.3 Requisitos No Funcionales

4.3.1 Requisitos de Tiempo de Latencia

R38. El sistema deberá validar los datos de usuario en un tiempo razonable, es decir, no deberá superar 1 segundo de media[22].

R39. El sistema deberá mostrar los resultados de una consulta en un tiempo razonable, es decir, de media no deberá superar los 5 segundos[23].

R40. El sistema deberá acceder a cada pantalla en un tiempo razonable, es decir, no deberá superar los 5 segundos de media.

[22] Los datos de conexión se almacenan en el terminal una vez se ha conectado el usuario al sistema por primera vez, por lo que la validación es muy rápida.

[23] Nótese que los tiempos que se establecen son algo elevados por los recursos limitados que tienen los dispositivos móviles, y porque, en su mayoría, estos tiempos van a depender de la cobertura y velocidad de conexión de la red en cada momento.

4.3.2 Requisitos de Usabilidad

R41. De tal forma que los usuarios estén lo más familiarizados posible con la aplicación, y previendo un posible cambio de terminal (y, consecuentemente, de plataforma), el sistema se asemejará al ya existente.

5. Análisis y Diseño de Alto Nivel

5.1 Decisiones de Diseño

5.1.1 Diseño de la Arquitectura

Para el desarrollo de este proyecto se ha optado por la utilización del entorno de desarrollo "Eclipse" y la instalación de la SDK de Android. Además, para hacer posible el desarrollo en Eclipse, se decide instalar el complemento de las herramientas de desarrollo de Android, ADT (de las siglas en inglés "Android Development Tools"), de tal manera que se pueda disponer de las herramientas de depuración y librerías necesarias.

Se recuerda al lector que el lenguaje de programación es Java debido a que es el único soportado por la arquitectura, además de XML para la definición del diseño de la interfaz de usuario.

Como gestor de bases de datos interno se emplea SQLite, para el que Android proporciona soporte a través de sus librerías, y mediante el cual un usuario de la aplicación puede mantener sus datos y recursos de manera local en el terminal.

Por otro lado, al tratarse este proyecto de una migración a otra plataforma de un proyecto previo ya existente, se verá, en el diseño detallado, que el modelo de datos, en su mayoría, no es propio sino que depende de un conjunto de servicios Web ya implementados.

5.1.2 Estructura de una aplicación sobre Android

Todas las aplicaciones desarrolladas para el sistema operativo Android siguen un framework específico a partir del cual se estructura el proyecto de la aplicación.

Cuando se crea un proyecto Android, el programador asigna un nombre para dicho proyecto, indicará el paquete que contendrá las clases principales, así como el nombre de la clase principal (nombre de la "actividad" principal) y el nombre de la aplicación tal y como quiere que aparezca en el emulador (o dispositivo) Android.

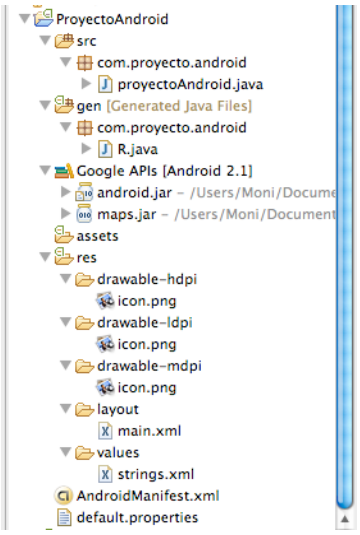


Ilustración 11 - Estructura de una aplicación Android

Como se puede apreciar en la ilustración anterior, el código de la aplicación se estructura de la siguiente manera:

- *Carpeta "src":* código fuente de la aplicación. Se sigue el mismo sistema de paquetes que en Java, en el que se crean paquetes que albergan clases con la misma funcionalidad.

  Se ha creado la clase principal de la aplicación "proyectoAndroid.java".
- *Carpeta "gen":* carpeta especial que contiene la clase "R.java". Se trata de una clase que no puede manipularse manualmente, sino que se genera automáticamente cada vez que el programador modifica los ficheros de recursos.

  Será el nexo de unión entre los recursos y los ficheros XML de interfaz, y las clases Java.
- *Librerías Android:* proporcionan la funcionalidad del sistema. Dependerán del SDK sobre el que se esté desarrollando la aplicación.
- *Carpeta "res":* Almacena, mediante un sistema de subcarpetas, los recursos que requiere la aplicación (imágenes, archivos XML del layout de la interfaz, ficheros XML multi-idioma, etc.).

- drawable: incluye todos los ficheros de imágenes. Se observan tres carpetas diferenciadas según la calidad.
- layout: ficheros XML que forman la interfaz de usuario.
- values: por defecto crea el fichero "string.xml" en el que se almacenarán todas las cadenas de texto que aparecerán en la aplicación.

  Además podrá contener ficheros con los valores para los atributos de la aplicación y/o sus estilos.

- *Fichero "AndroidManifest.xml":* fichero sin el cual no podrá construirse una aplicación Android. Contiene los parámetros de configuración: privilegios y permisos que se le asignan, actividad principal, definición del resto de actividades, etc.

5.1.2.1 Componentes Principales de una Aplicación Android[24]

Una vez comprendida la estructura de una aplicación Android, es conveniente explicar algunos de los componentes principales que puede instanciar y ejecutar según sus necesidades. Se trata de los siguientes cuatro componentes: activities, services, listeners y content providers.

- **Actividades**

  Una actividad (*activity*) es el elemento más utilizado en el desarrollo de una aplicación Android, ya que presenta una interfaz visual para una tarea determinada que puede realizar un usuario.

  Se debe tener en cuenta que una aplicación puede constar de una o varias actividades, pero siempre se marcará una de ellas como actividad principal (en el manifiesto) y se pasará de una a otra cuando la primera arranque la segunda.

  En cuanto a la disposición física de una actividad, ésta dispone de una ventana por defecto que podrá ocupar la pantalla completa del dispositivo, o ser una ventana flotante de menor tamaño.

[24] Se incluyen ejemplos de código de estos componentes en el Anexo A.

El contenido visual de una ventana depende de una jerarquía de vistas (objetos que derivan de la clase base "View"). Cada vista controla un espacio rectangular particular dentro de la ventana. Las vistas padre contienen y organizan la disposición de las vistas hijas. Las vistas hoja (situadas en la parte inferior de la jerarquía), "pintarán" en los rectángulos que controlan y responderán a las acciones de usuario dirigidas a ese espacio concreto. Por tanto, las vistas son el lugar en el que la interacción entre usuario y actividad tiene lugar.

Sabiendo que una actividad está compuesta por vistas, un cambio entre actividades, mediante la clase propia de Android *Intent*, supone un cambio entre vistas, en el que la vista actual queda relegada a un segundo plano (pausada) y se sitúa en una pila de historial de actividades para poder retornar a ella en caso necesario. En caso de que no se quiera retornar a esa actividad (por ejemplo en el caso de un formulario de solicitud de información), se podrá eliminar la actividad del historial (comando "finish()").

- **Servicios**

  Un servicio no tiene una interfaz de usuario, sino que ejecuta en segundo plano durante un periodo indefinido de tiempo. Los servicios implementan la clase básica de Android: "Service".

  A modo de ejemplo, se podría considerar el servicio que permite que una canción siga sonando aunque el usuario cierre el programa de reproducción de música (o cambie a otra actividad).

  En Android es posible conectarse/unirse a un servicio en curso (o iniciarlo si no está ejecutando) de manera que, mientras permanezca activa esa conexión, es posible comunicarse con el servicio a través de una interfaz que éste muestra.

- **Receptor de difusiones**

  Se trata de un componente encargado únicamente de recibir y reaccionar ante la llegada de difusiones y que extiende la clase básica "BroadcastReceiver".

  Un receptor no muestra una interfaz de usuario, pero pueden iniciar una actividad en respuesta a la información que reciban, o bien alertar al usuario a través del gestor de notificaciones ("NotificationManager").

  Muchas de las difusiones que pueden surgir lo harán a partir del código de sistema: aviso de batería baja o la toma de una fotografía. Sin embargo, una aplicación también puede iniciar difusiones para, por ejemplo, avisar a otras aplicaciones de que se ha obtenido la información solicitada.

- **Proveedores de Contenidos**

  Un proveedor de contenidos permite que otras aplicaciones obtengan acceso a un conjunto específico de datos de la aplicación actual. Esta información puede almacenarse en bases de datos SQLite, en el propio sistema de ficheros o en cualquier otro medio posible que el desarrollador considere apropiado.

  Extiende la clase base "ContentProvider" para implementar un conjunto estándar de métodos que permiten que otras aplicaciones obtengan y almacenen información del tipo que controla.

Se pueden encontrar ejemplos de código de estos componentes, así como del modo de activación de los mismos (ver siguiente apartado) en el Anexo A.

5.1.2.2 Activando Componentes: "Intent"

Los proveedores de contenidos se activan por medio de una petición de un "ContentResolver". Sin embargo, los otros tres componentes descritos en el apartado anterior se activan a través de mensajes asíncronos: *intents*.

Un *intent*, o intención, es un objeto de la clase "Intent" que alberga el contenido del mensaje que describe la acción que quiere llevar a cabo la aplicación. Posee dos partes fundamentales:

- La **acción** que se quiere realizar
- **Información adicional** necesaria para poder ejecutar la acción[25]

Métodos de activación de los distintos componentes:

- **Actividad:** se pasa un objeto *intent* a los métodos encargados de iniciar una actividad ("Context.startActivity()" o "Activity.startActivityForResult()").
  La actividad llamada podrá consultar el *intent* invocando el método "getIntent()" y obteniendo, por tanto, la información adicional asociada.
- **Servicio:** se pasa un objeto *intent* al método "Context.startService()" o al método "Context.bindService".
- **Difusión:** se pasa un *intent* a los distintos métodos de envío de difusiones, lo que hará que Android envíe el *intent* a todos los receptores interesados a través de su método "onReceive()".

5.1.2.3 Fichero "AndroidManifest"

Se trata de un fichero empaquetado en el fichero .apk[26] en el que la aplicación declara todos sus componentes. Sin este fichero, Android no sabrá qué componentes forman la aplicación y no podrá ejecutarlos.

Es un fichero XML estructurado que, además, indica las librerías contra las que necesita enlazarse la aplicación e identifica todos los permisos que se espera se concedan a la aplicación.

Se incluye un ejemplo de código en el Anexo B.

[25] Se indica en formato URI: http://tools.ietf.org/html/rfc3986
[26] Paquete resultante de construir la aplicación, que contiene el código, los ficheros, los recursos y el fichero AndroidManifest.

5.1.3 Elección del Protocolo de Comunicación entre Android y los Servicios Web

Dos opciones viables para llevar a cabo la conexión de una aplicación desarrollada para Android y una serie de servicios Web existentes son SOAP (Simple Object Access Protocol) y HTTP puro (HyperText Transfer Protocol), por ser lo suficientemente ligeros como para su uso en dispositivos móviles.

Otro protocolo, cada vez más utilizado, es REST. Sin embargo, se descartó por el escaso soporte que le proporciona Android, quien "prefiere" decantarse por comunicaciones a través de SOAP, y porque implicaría la reescritura de código en la parte servidora, ya existente, de la aplicación.

Antes de decantarse por uno de los protocolos en particular, se llevaron a cabo diversas pruebas de funcionamiento.

5.1.3.1 SOAP

Protocolo estándar que define cómo se llevará a cabo la comunicación entre dos objetos de diferentes procesos por medio del intercambio de mensajes XML. Deriva de un protocolo creado por David Winer en 1998 (XML-RPC), y fue creado por Microsoft e IBM, entre otros.

Asimismo, SOAP dispone de HTTP como protocolo de transporte.

A pesar de ser uno de los protocolos por excelencia utilizados en los servicios Web, a la hora de emplearlo para aplicaciones móviles (por naturaleza deben ser lo más ligeras posibles), se debe recurrir a una versión modificada de éste: kSOAP.

kSOAP no es más que la librería SOAP para clientes de servicios Web para entornos restringidos Java, como pueden ser los Applets y aplicaciones J2ME.

Los tres elementos clave en una comunicación SOAP son:

- SoapObject: Objeto dinámico simple empleado para construir mensajes de petición de servicio. Es lo que constituye el cuerpo de un "envelope" (o "sobre").
- SoapPrimitive: Clase empleada para encapsular tipos primitivos, que se representarán por medio de cadenas en la serialización XML.

- SoapSerializationEnvelope: mensaje que será enviado tanto como solicitud como por respuesta.

En cuanto al uso de este protocolo (kSOAP), se tenía la posibilidad de, o bien desarrollar una librería propia para el manejo de los mensajes, o bien emplear alguna librería ya desarrollada y disponibles con licencia gratuita.

Por tratarse este proyecto de un desarrollo bastante extenso, se optó por la segunda opción, probando varias alternativas tras una exhausta investigación de qué librerías ya desarrolladas por terceros se suponía eran las más adecuadas para su uso en Android.

- **Opción 1:** *"ksoap2-android-assembly-2.3-jar-with-dependencies.jar"*
  Proyecto desarrollado a partir de la librería estándar KSOAP2[27], incluyendo nuevas funcionalidades que proporcionan soporte para la plataforma Android. Publicada por Jorge Jiménez en el grupo de Google "Android Developers".

  Esta librería no terminaba de funcionar de manera fluida con la aplicación en desarrollo, ya que fallaba en la sentencia de creación del mensaje al elegir la versión a utilizar:

  ```
  SoapSerializationEnvelope envelope2 = new SoapSerializationEnvelope(SoapEnvelope.VER11);
  ```

  Se probó con todas las versiones posibles, pero no siempre se realizaba la comunicación con los servicios Web de manera correcta.

- **Opción 2:** *"Ksoap2-android-full-2.1.2.jar"* [28]
  Librería basada también en KSOAP2, pero a la que el desarrollador ha añadido dos nuevas clases: una para la conexión y otra para el transporte, que emplea Apache HttpComponents (incluidos en Android).

[27] http://sourceforge.net/projects/
[28] http://wiki.tuxpan.com:8069/android-soap/src-ksoap2-with-apache-http.zip

El uso de esta librería causaba una incompatibilidad en la aplicación y daba un error de falta de memoria en el entorno de desarrollo.

- **Opción 3:** *"ksoap-j2se-full-2.1.2.jar"* [29]
  Librería original, y sin ninguna modificación añadida, para la construcción de applets basados en SOAP.

  Ejemplo de construcción de un mensaje de solicitud con kSOAP:

```
private static final String NAMESPACE = "http://service.geobuddies/";
private static final String URL =
          "http://castor.dia.fi.upm.es:8080/GeoBuddies/UsuariosService";

private String SOAP_ACTION = null;
private String METHOD_NAME = null;

public ServiciosUsuarios()
{
    SOAP_ACTION = "http://service.geobuddies/";
}

public Usuario recuperarUsuario(int idUsuario)
{
    Usuario user = new Usuario();
    METHOD_NAME = "recuperarUsuario";

    // Probando ksoap y servicios web
    SoapObject request = new SoapObject(NAMESPACE, METHOD_NAME);
    request.addProperty("idUsuario", String.valueOf(idUsuario));
    HttpTransportSE androidHttpTransport = new HttpTransportSE(URL);
    SoapSerializationEnvelope envelope = new
                  SoapSerializationEnvelope(SoapEnvelope.VER11);
    envelope.dotNet = false;
    envelope.setOutputSoapObject(request);
    try
    {
        androidHttpTransport.call(SOAP_ACTION + METHOD_NAME, envelope);
    SoapObject resultsRequestSOAP = (SoapObject) envelope.bodyIn;

    SoapObject usuario = (SoapObject)
                  resultsRequestSOAP.getProperty("return");
    SoapObject direccion = (SoapObject) usuario.getProperty("direccion");
    // Creamos la dirección del usuario
    Direccion dir = new Direccion();
    dir.setCalle(direccion.getProperty("calle").toString());
    dir.setCiudad(direccion.getProperty("ciudad").toString());

    dir.setCp(Integer.parseInt(direccion.getProperty("cp").toString()));
    dir.setIdDireccion(Integer.parseInt(direccion.getProperty("idDireccion").
                  toString()));
    dir.setPais(direccion.getProperty("pais").toString());
    dir.setTelefono(direccion.getProperty("telefono").toString());
    // Creamos un usuario con estos datos
    user.setApellidos(usuario.getProperty("apellidos").toString());
    user.setDireccion(dir);
    user.setEmail(usuario.getProperty("email").toString());
```

[29] http://ksoap.objectweb.org/software/downloads/current/ksoap-j2se.zip

```
            user.setExtAvantar(usuario.getProperty("extAvantar").toString());

            user.setIdUsuario(Integer.parseInt(usuario.getProperty("idUsuario").
                        toString()));

            user.setLocalizar(Boolean.parseBoolean(usuario.getProperty("localizar").
                        toString()));
            user.setLogin(usuario.getProperty("login").toString());
            user.setNombre(usuario.getProperty("nombre").toString());
            user.setPassword(usuario.getProperty("password").toString());

            user.setTipo(Integer.parseInt(usuario.getProperty("tipo").toString()));

            }
            catch (ExceptionInInitializerError e)
            {
            System.out.println(e.getMessage());
            e.getCause();

            }
            catch(Exception e)
            {
            System.out.println(e.getMessage());
            }

            return user;
        }
```

5.1.3.2 HTTP

Es el protocolo por excelencia de Internet. Aunque se entiende, principalmente, como un medio para cargar páginas HTML en navegadores Web, HTTP es en realidad un protocolo de transporte genérico para cualquier tipo de datos. Es esta versatilidad la que hace del soporte de este protocolo una característica fundamental para el Perfil de Dispositivos de Información Móvil (MIDP – Mobile Information Device Profile).

Se trata de un protocolo de tipo solicitud-respuesta, en el que las solicitudes pueden ser de tipo GET o de tipo POST.

Para las pruebas realizadas, se optó por el método POST, aunque sin éxito, ya que no se consiguió establecer comunicación alguna con los servicios Web.

Ejemplo de construcción de un mensaje de solicitud con HTTP:

```
public static String callWebServiceMethod(int id)
{
   HttpClient httpClient = new DefaultHttpClient();
   HttpPost httpPost = new
       HttpPost("http://castor.dia.fi.upm.es:8080/GeoBuddies/UsuariosService?WSDL");
   String  webServiceXml = "";
   String  response = "";
```

```
    try
    {
       webServiceXml += "<?xml version=\"1.0\" encoding=\"utf-8\"?>";
    webServiceXml += "<soap:Envelope
              xmlns:soap=\"http://schemas.xmlsoap.org/soap/envelope/\">";
    webServiceXml += "<soap:Header/>";
    webServiceXml += "<soap:Body>";
    webServiceXml += "<recuperarUsuario xmlns=\"http://service.geobuddies/\">";
    webServiceXml += "<idUsuario>" + id + "</idUsuario>";
    webServiceXml += "</recuperarUsuario>";
    webServiceXml += "</soap:Body>";
    webServiceXml += "</soap:Envelope>";

       httpPost.setHeader("content-type","text/xml; charset=utf-8");
       httpPost.setHeader("SOAPAction", SOAP_ACTION);

       httpPost.setEntity(new StringEntity(webServiceXml));

       HttpResponse httpResponse = httpClient.execute(httpPost);

       response = EntityUtils.toString(httpResponse.getEntity());
    }
    catch(Exception ex)
    {
       Log.i("error", ex.getMessage());
    }

    return response;
}
```

5.1.3.3 Protocolo elegido

Finalmente, se escogió la opción 3 propuesta para el protocolo SOAP: *"ksoap-j2se-full-2.1.2.jar"*, ya que fue la única librería que permitió obtener respuesta del servidor al enviar mensajes SOAP de consulta.

Al realizar la llamada al servicio Web, el objeto resultado de la consulta quedaba alojado en el atributo "bodyIn" del "envelope". De esta forma, desde el código Java de la aplicación Android se podían consultar los valores de los distintos campos del objeto devuelto, tal y como se muestra en el extracto de código a continuación:

```
...
androidHttpTransport.call(SOAP_ACTION + METHOD_NAME, envelope);
SoapObject resultsRequestSOAP = (SoapObject) envelope.bodyIn;

SoapObject usuario = (SoapObject) resultsRequestSOAP.getProperty("return");
SoapObject direccion = (SoapObject) usuario.getProperty("direccion");
...
```

5.2 Diseño del Sistema

5.2.1 Estructura de la Aplicación

A continuación se muestra un diseño general modular que mantiene una idea de cómo se comunican los distintos módulos implementados entre sí:

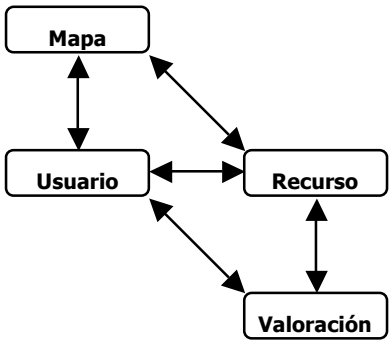


Ilustración 12 - Estructura modular del sistema

Seguidamente se puede observar la relación de los distintos módulos con las acciones que se han de poder realizar en cada uno de ellos:

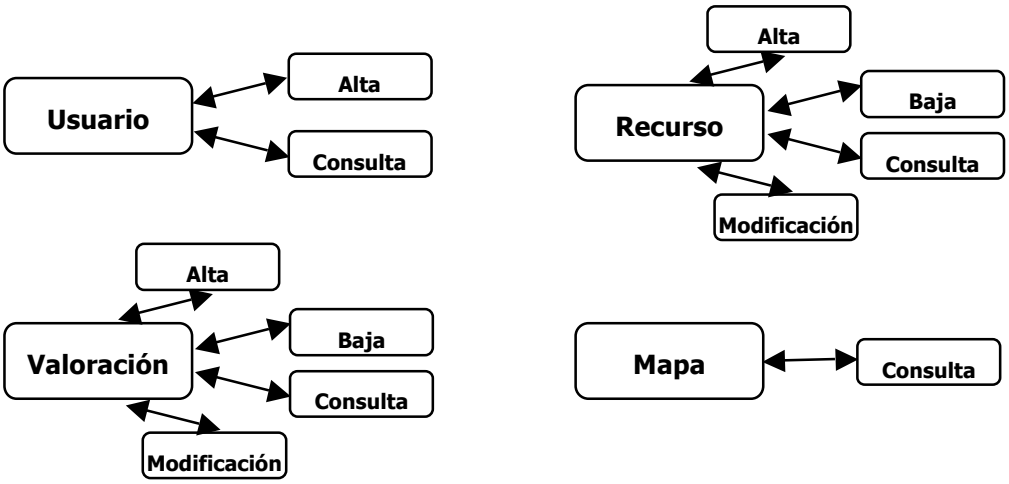


Ilustración 13 - Funcionalidades de los Módulos

5.2.2 Modelo del Dominio

Se muestran las clases conceptuales del dominio del problema y su relación.

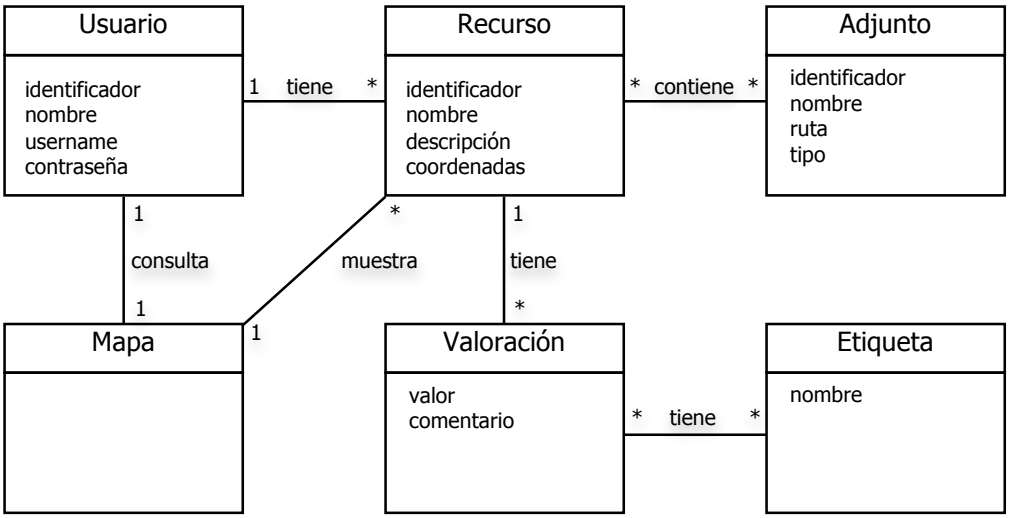


Ilustración 14 - Modelo Conceptual

5.2.3 Modelo de Casos de Uso

Se incluyen en el diseño los diagramas de casos de uso que recogen las acciones principales que un usuario (o actor) de la aplicación podrá llevar a cabo con la misma.

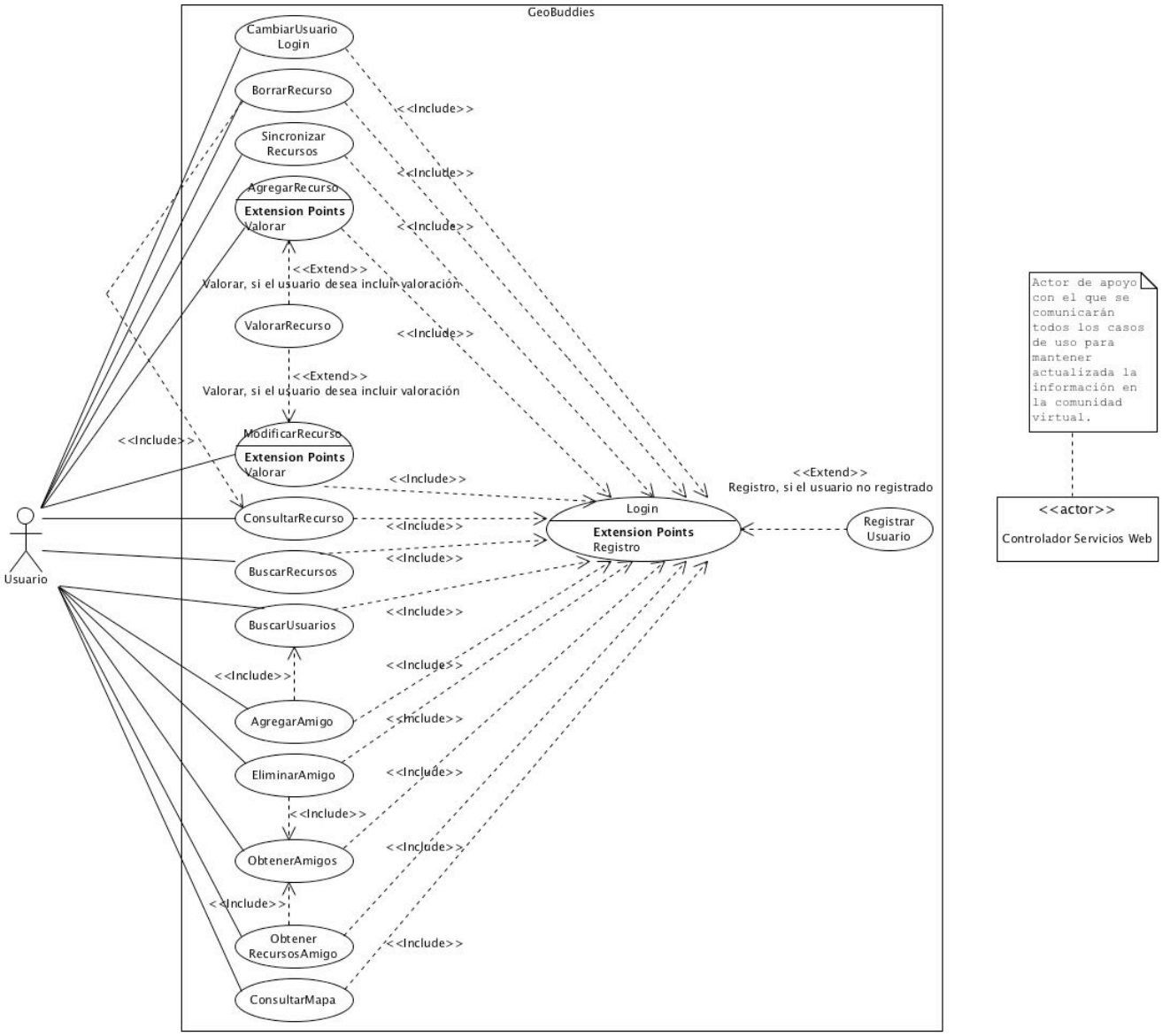


Ilustración 15 - Modelo de Casos de Uso

5.2.3.1 Detalle de los Casos de Uso

A continuación se detallan los casos de uso mostrados anteriormente.

Caso de uso UC1: RegistrarUsuario

Actor Principal: Usuario del dispositivo.

Personal involucrado e intereses:

- Usuario del dispositivo: quiere darse de alta en la aplicación para poder consultar y valorar los distintos recursos del Camino de Santiago.
- Desarrollador de la aplicación: quiere que se registre un número elevado de usuarios para crear una comunidad virtual de éxito para satisfacer las necesidades de los usuarios.

Precondiciones: La aplicación está arrancada en el terminal móvil.

Postcondiciones: Se registra el usuario correctamente. Se genera una confirmación de alta.

Condición de Entrada: El usuario solicita el servicio de Registro.

Punto de Extensión: Registro, en caso de uso UC2 Login.

Escenario principal de éxito:

1. El usuario solicita registrarse en la aplicación.
2. El sistema muestra un formulario solicitando los datos de registro.
3. El usuario introduce sus datos y pulsa "aceptar".
4. El sistema verifica que los datos son correctos y registra al usuario. Muestra mensaje de confirmación de alta.

Flujos alternativos:

*a. El usuario solicita cancelar la acción:

1. En cualquier momento el usuario solicita cancelar la acción.
2. El sistema rechaza la operación.

3a. Comprobar el nombre de usuario:

1. El usuario solicita comprobar la disponibilidad del nombre de usuario seleccionado.
2. El sistema verifica la disponibilidad e indica que es correcto.
   2a. El sistema detecta que ese nombre de usuario ya existe:
      1. El sistema informa del error al usuario y solicita nuevo nombre de usuario.

4a. Los datos no son correctos:

1. El Sistema comprueba que hay datos erróneos (disponibilidad del nombre de usuario, contraseña mal introducida, etc.) y muestra el error al usuario rechazando el registro.

Caso de uso UC2: Login

Actor Principal: Usuario del dispositivo.

Personal involucrado e intereses:

- Usuario del dispositivo: quiere iniciar su sesión en la aplicación para poder consultar y valorar los distintos recursos del Camino de Santiago.
- Desarrollador de la aplicación: quiere que el inicio de sesión sea lo más cómodo posible para el usuario, haciendo que éste pueda iniciarla sin necesidad de cobertura/conexión.

Precondiciones: La aplicación está arrancada en el terminal móvil.

El usuario está registrado en la aplicación.

Postcondiciones: El usuario inicia sesión y se le muestra la pantalla principal.

Punto de Extensión: Registro, paso 1.

Escenario principal de éxito:

1. El sistema solicita los datos de sesión.
2. El usuario se identifica.
3. El sistema autentica la identidad y muestra la pantalla de inicio.

Flujos alternativos:

*a. El usuario solicita cancelar la acción:

1. En cualquier momento el usuario solicita cancelar la acción.
2. El sistema rechaza la operación.

3a. La identidad del usuario no se corresponde con la almacenada en el terminal:

1. El sistema comunica el error y ofrece al usuario un nuevo intento de inicio de sesión.

Caso de uso UC3: CambiarUsuarioLogin

Actor Principal: Usuario del dispositivo.

Personal involucrado e intereses:

- Usuario del dispositivo: quiere asociar una cuenta de usuario distinto al terminal.
- Desarrollador de la aplicación: quiere ofrecer la posibilidad de que más de un usuario pueda conectarse desde el mismo terminal.

Precondiciones: La aplicación está arrancada en el terminal móvil.

El usuario está registrado y ha iniciado sesión en la aplicación.

El nuevo usuario a vincular también debe estar registrado.

Postcondiciones: El usuario vinculado al terminal ha sido cambiado satisfactoriamente.

Escenario principal de éxito:

1. El usuario solicita el cambio de usuario de inicio de sesión.
2. El sistema solicita los datos de sesión del nuevo usuario.
3. El usuario introduce los datos.
4. El sistema valida los datos y asocia el nuevo usuario al dispositivo.

Flujos alternativos:

*a. El usuario solicita cancelar la acción:

1. En cualquier momento el usuario solicita cancelar la acción.
2. El sistema rechaza la operación.

4a. Los datos introducidos no son correctos:

2. El sistema comunica el error y ofrece al usuario otro intento de asociación de nuevo usuario.

Caso de uso UC4: AgregarRecurso

Actor Principal: Usuario del dispositivo.

Personal involucrado e intereses:

- Usuario del dispositivo: quiere crear un nuevo recurso de un punto de interés.
- Desarrollador de la aplicación: quiere que el registro del nuevo recurso sea rápido y posible incluso sin conectividad.

Precondiciones: La aplicación está arrancada en el terminal móvil.

El usuario está registrado y ha iniciado sesión en la aplicación.

Postcondiciones: El recurso se ha registrado con éxito y está disponible localmente.

Punto de Extensión: Valorar, paso 3.

Escenario principal de éxito:

1. El usuario solicita agregar un nuevo recurso.
2. El sistema solicita la información asociada al recurso.
3. El usuario introduce los datos.
4. El sistema registra el recurso.

Flujos alternativos:

*a. El usuario solicita cancelar la acción:

1. En cualquier momento el usuario solicita cancelar la acción.
2. El sistema rechaza la operación.

3a. El usuario adjunta una foto al recurso:

1. El usuario solicita adjuntar una imagen.
2. El sistema muestra el sistema de ficheros solicitando escoger una imagen.
3. El usuario selecciona la imagen deseada.
4. El sistema procesa la ruta del fichero y la añade al recurso.

3b. El usuario adjunta una archivo de ruta al recurso:

1. El usuario solicita adjuntar un mapa de ruta.
2. El sistema muestra el sistema de ficheros solicitando escoger una ruta.
3. El usuario selecciona la ruta deseada.
4. El sistema procesa la ruta del fichero y la añade al recurso.

Caso de uso UC5: SincronizarRecursos

Actor Principal: Usuario del dispositivo.

Personal involucrado e intereses:

- Usuario del dispositivo: quiere sincronizar los recursos de su terminal con los que tiene en la comunidad virtual.
- Desarrollador de la aplicación: quiere ofrecer al usuario la posibilidad de sincronizar sus recursos en el momento más adecuado.

Precondiciones: La aplicación está arrancada en el terminal móvil.

El usuario está registrado y ha iniciado sesión en la aplicación.

Postcondiciones: Se han sincronizado todos los recursos pendientes en la comunidad virtual.

Escenario principal de éxito:

1. El usuario solicita sincronizar sus recursos.
2. El sistema verifica los recursos modificados y los registra en la comunidad virtual.

Flujos alternativos:

*a. El usuario solicita cancelar la acción:

1. En cualquier momento el usuario solicita cancelar la acción.
2. El sistema rechaza la operación.

2a. No hay conectividad:

1. El sistema informa del error y rechaza la petición.

Caso de uso UC6: BorrarRecurso

Actor Principal: Usuario del dispositivo.

Personal involucrado e intereses:

- Usuario del dispositivo: quiere borrar uno de sus recursos.
- Desarrollador de la aplicación: quiere ofrecer al usuario la posibilidad de borrar sus recursos para dejar de compartirlos con la comunidad.

Precondiciones: La aplicación está arrancada en el terminal móvil.

El usuario está registrado y ha iniciado sesión en la aplicación.

Postcondiciones: Se ha borrado el recurso solicitado de la aplicación.

Escenario principal de éxito:

1. El usuario solicita borrar un recurso.
2. El sistema da de baja el recurso.

Flujos alternativos:

*a. El usuario solicita cancelar la acción:

1. En cualquier momento el usuario solicita cancelar la acción.
2. El sistema rechaza la operación.

1a. Buscar un recurso: incluye *ConsultarRecurso*.

2a. No hay conectividad:

1. El sistema informa del error y rechaza la petición.

Caso de uso UC7: ValorarRecurso

Actor Principal: Usuario del dispositivo.

Personal involucrado e intereses:

- Usuario del dispositivo: quiere añadir una valoración a un recurso.
- Desarrollador de la aplicación: quiere ofrecer al usuario la posibilidad de incluir un valor añadido a los recursos de la comunidad por medio de valoraciones.

Precondiciones: La aplicación está arrancada en el terminal móvil.

El usuario está registrado y ha iniciado sesión en la aplicación.

Postcondiciones: Se ha añadido una valoración al recurso seleccionado.

Condición de Entrada: El usuario quiere valorar un recurso.

Punto de Extensión: Valorar en caso de uso UC4 AgregarRecurso.

Valorar en caso de uso UC8 ModificarRecurso.

Escenario principal de éxito:

1. El usuario solicita valorar el recurso seleccionado.
2. El sistema pide al usuario que introduzca su valoración.

3. El usuario inserta una valoración numérica.
4. El sistema registra la modificación del recurso.

Flujos alternativos:

*a. El usuario solicita cancelar la acción:

1. En cualquier momento el usuario solicita cancelar la acción.
2. El sistema rechaza la operación.

Caso de uso UC8: ModificarRecurso

Actor Principal: Usuario del dispositivo.

Personal involucrado e intereses:

- Usuario del dispositivo: quiere modificar algún recurso de la comunidad.
- Desarrollador de la aplicación: quiere que los recursos sean dinámicos y puedan modificarse en el tiempo.

Precondiciones: La aplicación está arrancada en el terminal móvil.

El usuario está registrado y ha iniciado sesión en la aplicación.

Postcondiciones: El recurso se ha modificado con éxito.

Punto de Extensión: Valorar, paso 3.

Escenario principal de éxito:

1. El usuario solicita modificar un recurso.
2. El sistema solicita la información a modificar.
3. El usuario introduce los datos.
4. El sistema registra el recurso.

Flujos alternativos:

*a. El usuario solicita cancelar la acción:

1. En cualquier momento el usuario solicita cancelar la acción.
2. El sistema rechaza la operación.

Caso de uso UC9: ConsultarRecurso

Actor Principal: Usuario del dispositivo.

Personal involucrado e intereses:

- Usuario del dispositivo: quiere consultar los detalles de un recurso.
- Desarrollador de la aplicación: quiere que los usuarios puedan consultar los detalles de los recursos.

Precondiciones: La aplicación está arrancada en el terminal móvil.

El usuario está registrado y ha iniciado sesión en la aplicación.

Postcondiciones: Se muestra la información asociada al recurso.

Escenario principal de éxito:

1. El usuario solicita consultar un recurso.
2. El sistema muestra la información asociada al recurso.

Flujos alternativos:

*a. El usuario solicita cancelar la acción:

1. En cualquier momento el usuario solicita cancelar la acción.
2. El sistema rechaza la operación.

Caso de uso UC10: BuscarRecursos

Actor Principal: Usuario del dispositivo.

Personal involucrado e intereses:

- Usuario del dispositivo: quiere buscar los recursos de una determinada índole.
- Desarrollador de la aplicación: quiere que los usuarios puedan acceder a los recursos de una forma fácil y ordenada.

Precondiciones: La aplicación está arrancada en el terminal móvil.

El usuario está registrado y ha iniciado sesión en la aplicación.

Postcondiciones: Se muestra una lista con los resultados de la búsqueda.

Escenario principal de éxito:

1. El usuario solicita buscar recursos.
2. El sistema pide al usuario que indique los criterios de búsqueda.
3. El usuario indica dichos criterios.
4. El sistema procesa los datos y muestra los recursos que atienden a esos criterios.

Flujos alternativos:

*a. El usuario solicita cancelar la acción:

1. En cualquier momento el usuario solicita cancelar la acción.
2. El sistema rechaza la operación.

4a. No existen resultados:

1. El sistema indica este hecho y finaliza la operación.

Caso de uso UC11: BuscarUsuarios

Actor Principal: Usuario del dispositivo.

Personal involucrado e intereses:

- Usuario del dispositivo: quiere buscar a un usuario determinado en la comunidad virtual.
- Desarrollador de la aplicación: quiere que los usuarios puedan mantener una relación de contactos preferidos.

Precondiciones: La aplicación está arrancada en el terminal móvil.

El usuario está registrado y ha iniciado sesión en la aplicación.

Postcondiciones: Se muestra una lista con los resultados de la búsqueda.

Escenario principal de éxito:

1. El usuario solicita buscar otro usuario.
2. El sistema pide al usuario que seleccione un criterio de búsqueda y un texto para buscar.
3. El usuario selecciona un criterio, introduce un parámetro de búsqueda e inicia la búsqueda.
4. El sistema procesa los datos y muestra los usuarios que cumplen el criterio.

Flujos alternativos:

*a. El usuario solicita cancelar la acción:

1. En cualquier momento el usuario solicita cancelar la acción.
2. El sistema rechaza la operación.

4a. No existen resultados:

1. El sistema indica este hecho y finaliza la operación.

Caso de uso UC12: AgregarAmigo

Actor Principal: Usuario del dispositivo.

Personal involucrado e intereses:

- Usuario del dispositivo: quiere agregar un usuario de la comunidad a su lista de favoritos.
- Desarrollador de la aplicación: quiere que los usuarios puedan mantener una relación de contactos preferidos.

Precondiciones: La aplicación está arrancada en el terminal móvil.

El usuario está registrado y ha iniciado sesión en la aplicación.

Postcondiciones: Se muestra la relación de amigos del usuario del dispositivo, entre los que se encontrará el usuario recién agregado.

Escenario principal de éxito:

1. El usuario solicita agregar un usuario.
2. El sistema solicita confirmación para agregar el nuevo amigo.
3. El usuario confirma la acción.
4. El sistema procesa la petición y agrega el usuario seleccionado como amigo del usuario del dispositivo.

Flujos alternativos:

*a. El usuario solicita cancelar la acción:

1. En cualquier momento el usuario solicita cancelar la acción.
2. El sistema rechaza la operación.

1a. Buscar un amigo: incluye *BuscarUsuarios*.

Caso de uso UC13: EliminarAmigo

Actor Principal: Usuario del dispositivo.

Personal involucrado e intereses:

- Usuario del dispositivo: quiere eliminar a un usuario determinado de su lista de contactos favoritos.
- Desarrollador de la aplicación: quiere que los usuarios puedan gestionar de manera sencilla su lista de contactos favoritos.

Precondiciones: La aplicación está arrancada en el terminal móvil.

El usuario está registrado y ha iniciado sesión en la aplicación.

Postcondiciones: Se muestra un mensaje de confirmación y la relación de amigos del usuario del dispositivo, entre los que ya no se encontrará el usuario recién eliminado.

Escenario principal de éxito:

1. El usuario solicita eliminar un contacto.
2. El sistema solicita una confirmación al usuario.
3. El usuario confirma la acción.
4. El sistema procesa la petición y elimina el usuario seleccionado de la lista de amigos del usuario del dispositivo.

Flujos alternativos:

*a. El usuario solicita cancelar la acción:

1. En cualquier momento el usuario solicita cancelar la acción.
2. El sistema rechaza la operación.

1a. Consultar Amigos: incluye *ObtenerAmigos*.

Caso de uso UC14: ObtenerAmigos

Actor Principal: Usuario del dispositivo.

Personal involucrado e intereses:

- Usuario del dispositivo: quiere obtener un listado de sus amigos de la comunidad virtual.
- Desarrollador de la aplicación: quiere que los usuarios puedan gestionar de manera sencilla su lista de contactos.

Precondiciones: La aplicación está arrancada en el terminal móvil.

El usuario está registrado y ha iniciado sesión en la aplicación.

Postcondiciones: Se muestra la relación de amigos del usuario del dispositivo.

Escenario principal de éxito:

1. El usuario solicita consultar su lista de contactos.
2. El sistema procesa la petición y muestra la lista.

Flujos alternativos:

*a. El usuario solicita cancelar la acción:

1. En cualquier momento el usuario solicita cancelar la acción.
2. El sistema rechaza la operación.

2a. No existen resultados:

1. El sistema indica este hecho y finaliza la operación.

Caso de uso UC15: ObtenerRecursosAmigo

Actor Principal: Usuario del dispositivo.

Personal involucrado e intereses:

- Usuario del dispositivo: quiere consultar los recursos de uno de sus contactos.
- Desarrollador de la aplicación: quiere que los usuarios puedan acceder rápidamente a los recursos de los usuarios que más le interesen.

Precondiciones: La aplicación está arrancada en el terminal móvil.

El usuario está registrado y ha iniciado sesión en la aplicación.

Postcondiciones: Se muestra la relación de recursos del usuario seleccionado.

Escenario principal de éxito:

1. El usuario solicita ver los recursos de un contacto.
2. El sistema muestra la lista de recursos del usuario seleccionado.

Flujos alternativos:

*a. El usuario solicita cancelar la acción:

1. En cualquier momento el usuario solicita cancelar la acción.
2. El sistema rechaza la operación.

1a. Consultar Amigos: incluye *ObtenerAmigos*.

Caso de uso UC16: ConsultarMapa

Actor Principal: Usuario del dispositivo.

Personal involucrado e intereses:

- Usuario del dispositivo: quiere consultar sus recursos en el mapa.
- Desarrollador de la aplicación: quiere que los usuarios puedan acceder a una representación gráfica de sus recursos.

Precondiciones: La aplicación está arrancada en el terminal móvil.

El usuario está registrado y ha iniciado sesión en la aplicación.

Postcondiciones: Se muestra la relación de recursos del usuario geoposicionados en un mapa.

Escenario principal de éxito:

1. El usuario solicita acceder al mapa.
2. El sistema muestra el mapa con los recursos del usuario. Posición inicial: la del usuario.

Flujos alternativos:

*a. El usuario solicita cancelar la acción:

1. En cualquier momento el usuario solicita cancelar la acción.
2. El sistema rechaza la operación.

2a. GPS inactivo:

1. El sistema detecta que el GPS del dispositivo está inactivo.
2. El sistema pide confirmación para activar la localización por GPS.
3. El usuario confirma.

3a. El usuario rechaza la petición de activar el GPS:

1. El usuario rechaza la petición.
2. El sistema muestra el mapa centrado en Santiago de Compostela.

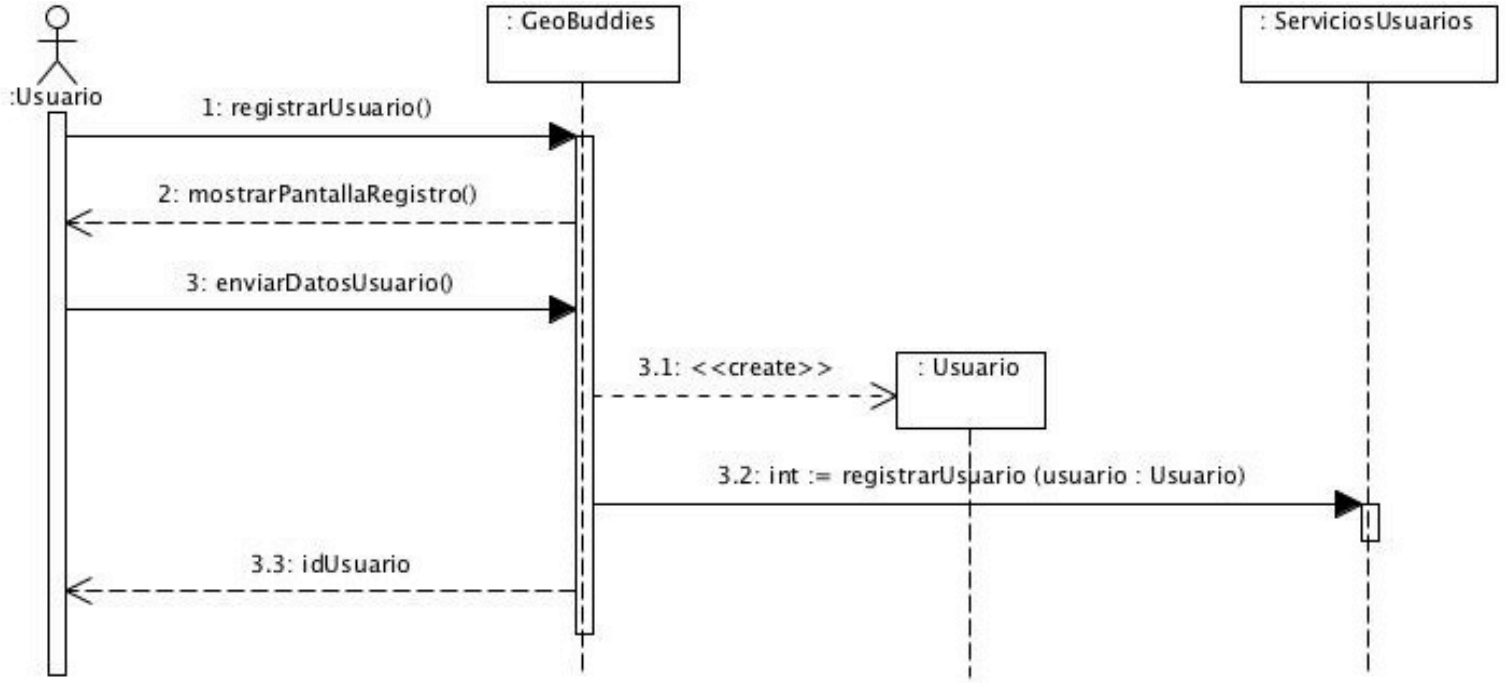
4. El sistema muestra el mapa centrado en la ubicación del usuario.

5.2.4 Diagramas de Secuencia

Se muestran los diagramas de secuencia de las operaciones principales disponibles desde la aplicación.

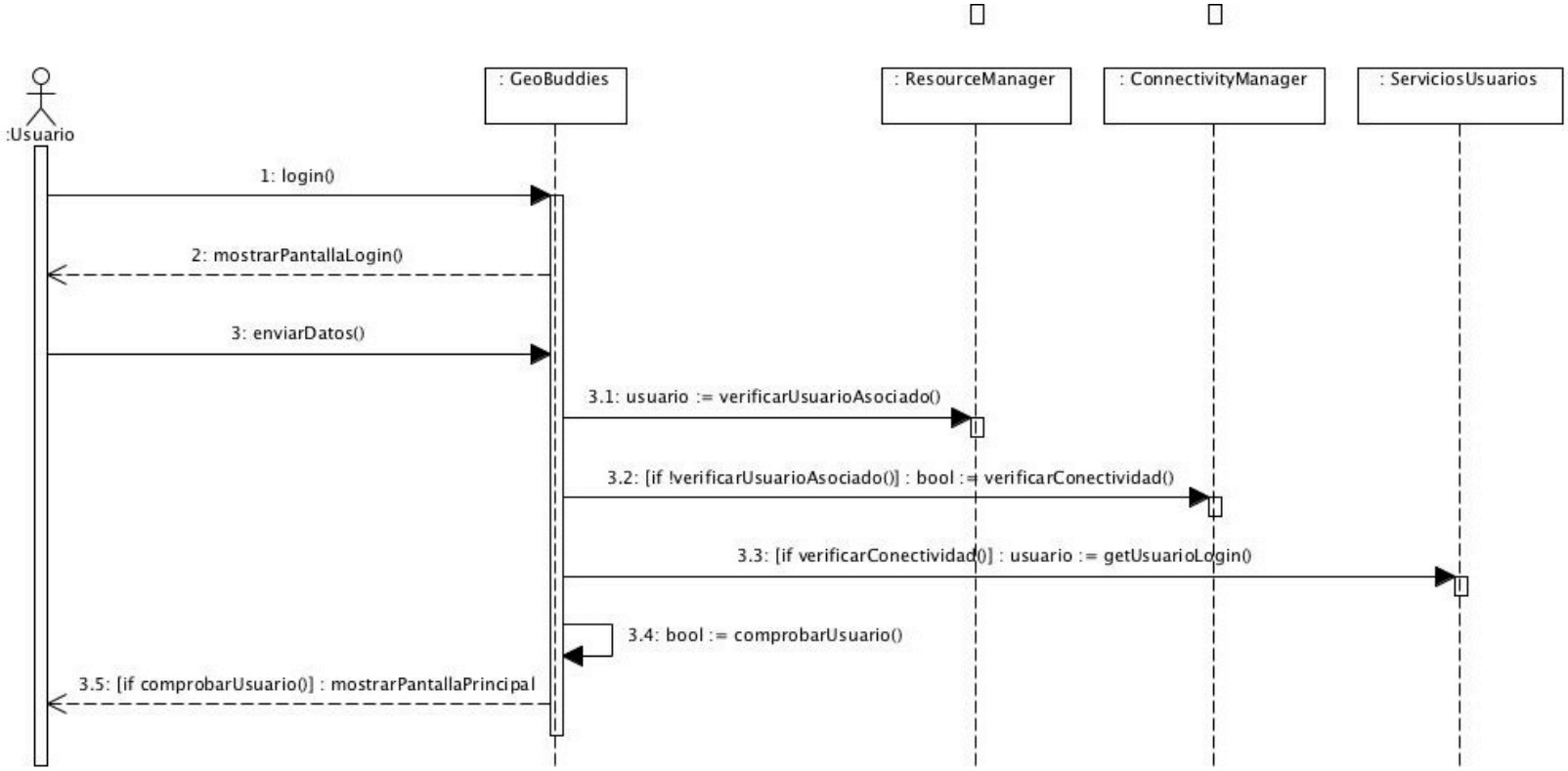
5.2.4.1 Registro de Usuario

El usuario de la aplicación solicitará darse de alta en la comunidad virtual. Acto seguido, se le mostrará una pantalla con un formulario de registro que deberá enviar para completar el alta. Se crea un objeto "Usuario", con los datos necesarios, que se enviará en el mensaje SOAP al servicio Web "registrarUsuario".



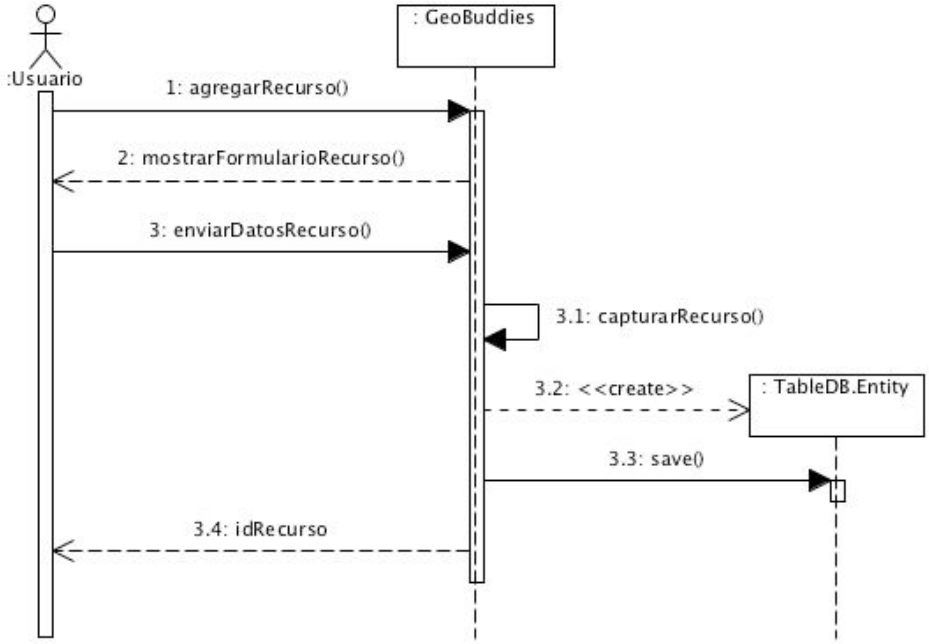
5.2.4.2 Inicio de Sesión

El usuario de la aplicación solicita iniciar sesión en la aplicación, que mostrará un formulario de inicio de sesión. En caso de que exista un perfil de usuario asociado a la aplicación, se comprobará si los datos introducidos por el usuario coinciden con los almacenados en el terminal. En caso de que no exista perfil, la aplicación comprobará si hay conectividad para comparar los datos introducidos con los de la BD de la comunidad virtual.



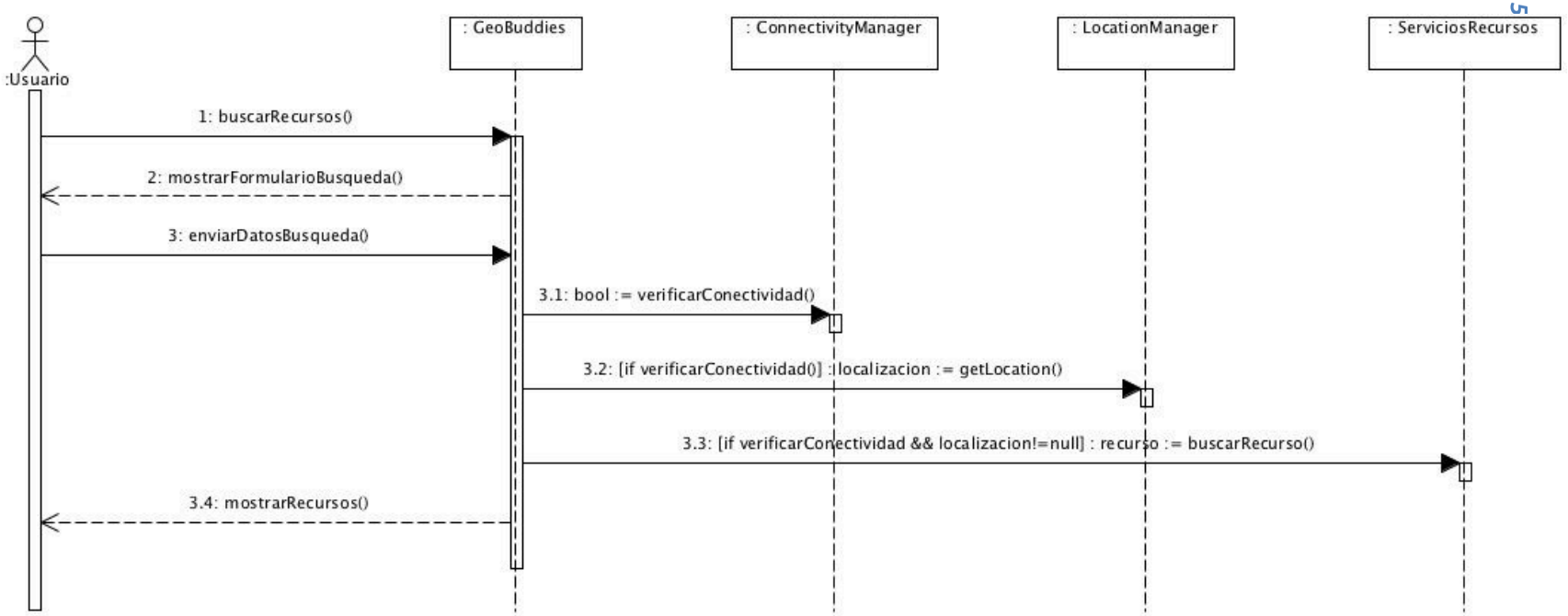
5.2.4.3 Agregar Recurso

El usuario del dispositivo introducirá los datos del recurso que quiere dar de alta y el sistema lo registrará en la base de datos local de la aplicación para una posterior sincronización a través de los servicios Web.



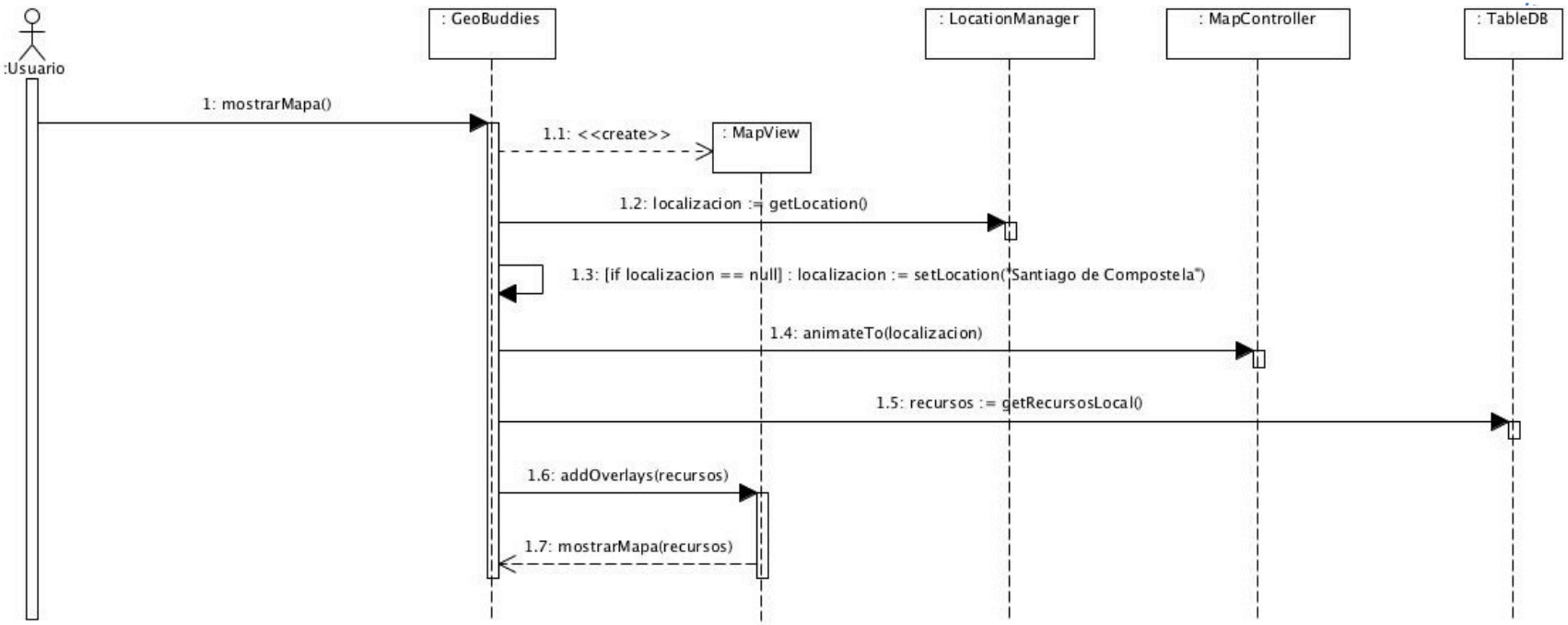
5.2.4.4 Inicio de Sesión

El usuario de la aplicación solicita iniciar sesión en la aplicación, que mostrará un formulario de inicio de sesión. En caso de que exista un perfil de usuario asociado a la aplicación, se comprobará si los datos introducidos por el usuario coinciden con los almacenados en el terminal. En caso de que no exista perfil, la aplicación comprobará si hay conectividad para comparar los datos introducidos con los de la BD de la comunidad virtual.



5.2.4.5 Mostrar Mapa

Si el sistema es capaz de obtener la localización geográfica del usuario, centrará el mapa en este punto. En caso contrario, lo centrará en Santiago de Compostela (lugar seleccionado por defecto). Además, sobre el mapa se mostrarán los recursos del usuario para que pueda gestionarlos más fácilmente



6 Diseño Detallado

6.1 Estructura de la Interfaz de Usuario

Se ha estructurado la aplicación de manera que se asemeje a la aplicación ya existente para fomentar la usabilidad. Asimismo, el acceso a las distintas secciones de la aplicación se integra con las diferentes posibilidades que ofrecen los terminales con sistema operativo Android.

El sistema mostrará los datos a modo de menús de opciones, menús secundarios, listas de resultados y mapas.

Cuando un usuario instala y arranca la aplicación por primera vez, se encuentra con una pantalla de inicio de sesión, en la que deberá registrarse (en caso de no haberlo hecho ya) para poder iniciar una sesión que le permita manejar sus recursos.

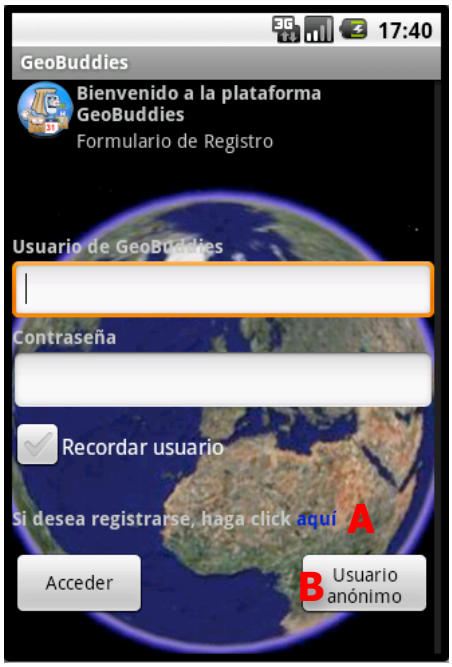


Ilustración 16 - Pantalla de inicio de sesión

Se puede apreciar una casilla de validación que permitirá al usuario recordar sus datos de inicio de sesión para que, al arrancar la aplicación nuevamente, ésta valide el usuario automáticamente y pase a la pantalla principal. Nótese que, de ser este el caso, la única forma de restituir la aplicación para que vuelva a solicitar la validación de la contraseña será desde el panel de preferencias.

En caso de que un usuario no se haya registrado previamente en la comunidad virtual, tendrá dos opciones:

- La primera, registrarse a través del link mostrado. (**A** en la imagen).
- La segunda, acceder como usuario anónimo. Esta última, opción muy útil en caso de querer acceder a la aplicación, para agregar un recurso por ejemplo, en un instante en el que el dispositivo no dispone de cobertura. (**B** en la imagen).

Si el usuario decide registrarse, la aplicación le presentará un formulario de registro como el siguiente:

Ilustración 17 - Formulario de registro

Este panel permitirá al usuario, además, verificar la disponibilidad del nombre de usuario de su elección en la comunidad virtual.

En caso de que el usuario decida entrar como usuario anónimo en la aplicación, la pantalla principal será igual que en el caso en que se identifique con la excepción de que el usuario no podrá acceder a sus recursos. Sí tendrá la oportunidad, sin embargo, de añadir nuevos recursos que, posteriormente, podrá sincronizar asociando una cuenta de usuario válida.

6.1.1 Pantalla principal

Una vez iniciada la sesión, el usuario accede a la pantalla principal de la aplicación, desde la cual podrá navegar a todas las distintas secciones, llevando a cabo, por tanto todas las acciones permitidas.

Ilustración 18 - Pantalla principal de la interfaz de usuario

Se observan tres secciones fundamentales en las que queda dividida la interfaz principal:

1. Acceso a "Mis Recursos"

   Permite al usuario visualizar los recursos que ha dado de alta y consultar los detalles.

2. Acceso a la "Comunidad" de amigos

   Permite al usuario llevar el control de sus usuarios favoritos/amigos: agregar, eliminar y consultar sus recursos.

3. Barra de menú dinámica

   Se trata de una barra de herramientas que proporciona acceso directo a la búsqueda de recursos, dar de alta nuevos recursos, comunidad, mapa, datos y preferencias. Además, proporciona una manera ordenada de cerrar la aplicación[30].

6.1.2 Mis Recursos

Se presentan los recursos del usuario en modo lista. Como se puede observar, cada recurso viene acompañado de una pequeña imagen que ayuda a identificar su origen (si se trata de una imagen, una ruta, un punto de interés, etc.).

Si el usuario quiere ver el recurso más detalladamente, sólo tendrá que seleccionarlo para abrir la pantalla de información.

Ilustración 19 - Visualización de "Mis Recursos"

[30] Garantiza el cierre ordenado de las distintas actividades que componen la aplicación. Sin embargo, Android permite a sus usuarios salir de las aplicaciones por medió del botón "atrás", aunque esto no las cierra, sino que las traslada a un segundo plano.

El lector podrá apreciar que, cuando el recurso es una ruta, o tiene una ruta asociada, aparece un nuevo icono en los detalles que permiten visualizarla en un mapa embebido en la aplicación. Dicho mapa, como se ve en las siguientes imágenes, permitirá la manipulación de la imagen: zoom y navegación.

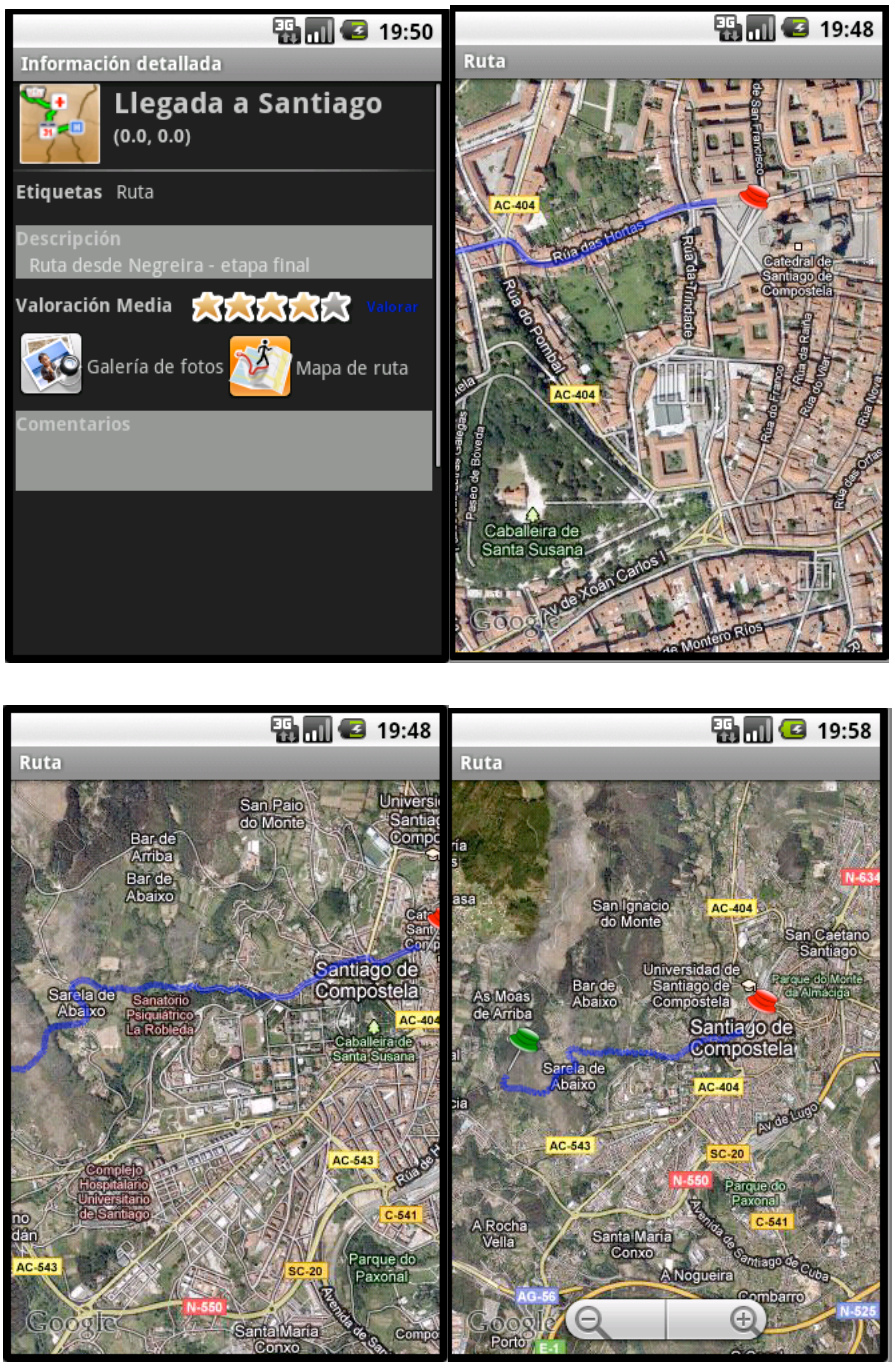


Ilustración 20 - Visualización de una ruta sobre el mapa

6.1.3 Comunidad de Amigos

Del mismo modo que con los recursos, esta pantalla permite la visualización, en modo lista, de los amigos que tiene el usuario. Al seleccionar un determinado amigo, se mostrarán sus recursos, pudiéndose visualizar tal y como ya se ha explicado anteriormente.

Ilustración 21 - Recursos de los "amigos"

Manteniendo la operativa de un número elevado de aplicaciones basadas en Android, para agregar o eliminar amigos, se emplean la tecla de menú y la presión larga sobre el elemento, respectivamente.

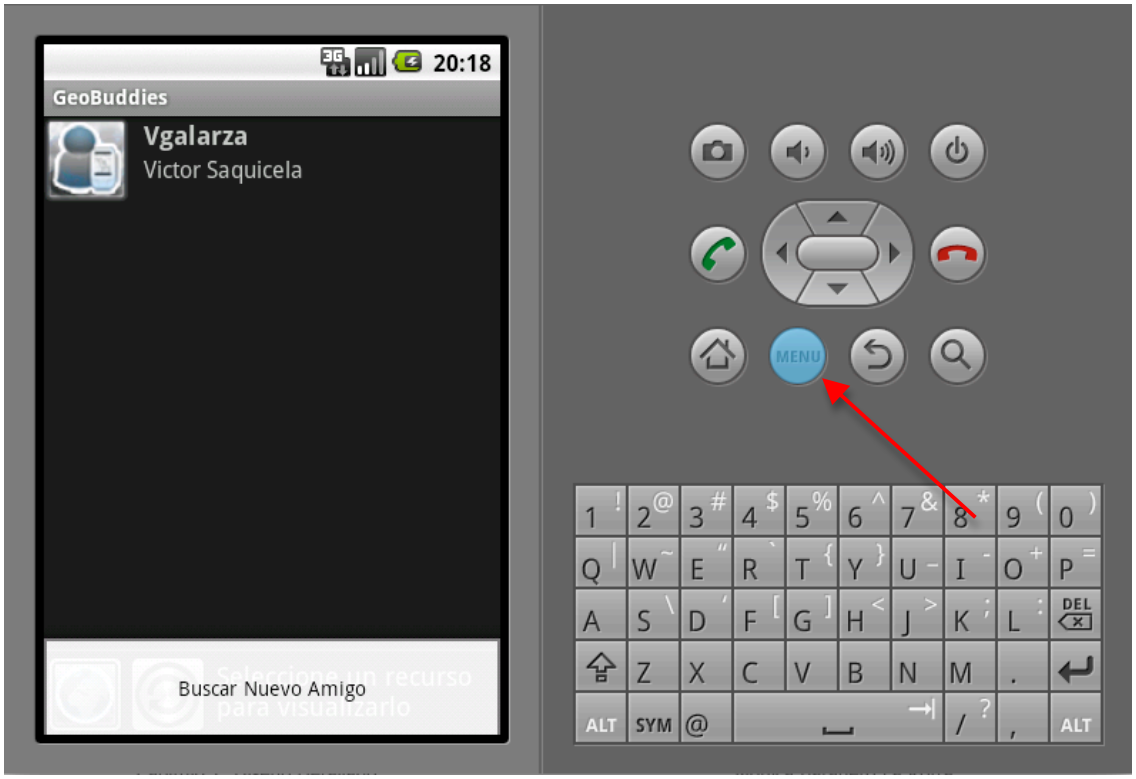


Ilustración 22 - Agregar Amigo

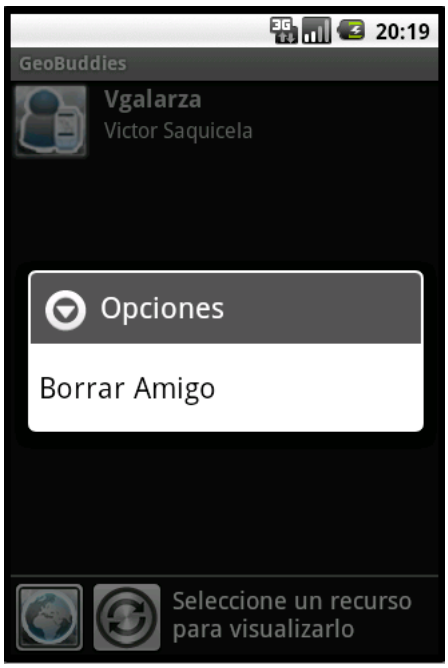


Ilustración 23 - Eliminar Amigo

6.1.4 Barra de Menú

La barra de menú se ha diseñado integrando en la pantalla principal un objeto común de Android: la galería. Es un menú deslizante en el que el elemento que queda en el centro (se ha decidido incorporar una etiqueta en la parte inferior a modo de ayuda al usuario) es el que se selecciona.

Ilustración 24 - Barra de Menú Principal

6.1.5 Menú de Preferencias

Tal como ocurre en un gran número de aplicaciones, existe un menú de preferencias basado en una clase específica de Android. Se trata del objeto XML "preferences.xml", para el que existen elementos predefinidos que facilitan la incorporación de selectores, descripciones y grupos, típicos de un menú de opciones:

- PreferenceScreen: una pantalla de preferencias.
- PreferenceCategory: un menú/categoría dentro de la pantalla de preferencias.
- ListPreference: muestra un icono de menú desplegable.
- CheckBoxPreference: muestra una casilla de activación.

A continuación se muestra la estructura que se puede conseguir con las diferentes vistas disponibles, además de una visión general del panel de preferencias de la aplicación en desarrollo.

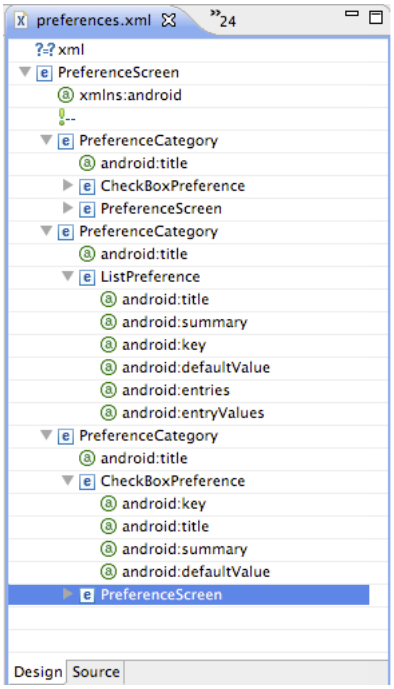


Ilustración 25 -Fichero XML de Preferencias

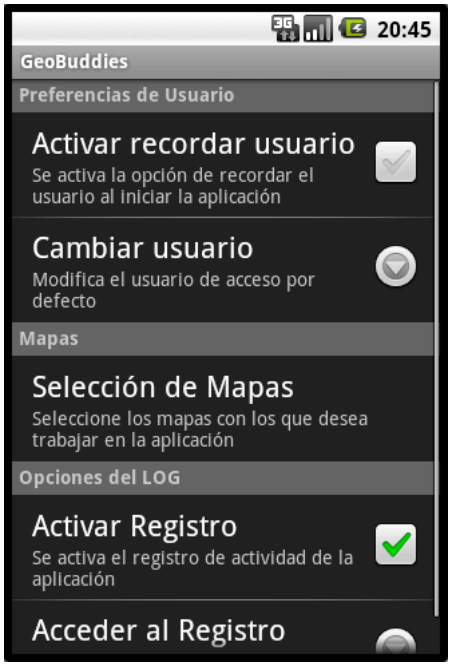


Ilustración 26 - Menú Ajustes

Todas las operaciones mencionadas anteriormente podrán verse en más detalle durante la descripción de las pruebas propuestas para la interfaz de usuario.

6.2 Modelo de Datos Local

Durante el desarrollo de la aplicación se empleará una base de datos local para almacenar los recursos del usuario. El sistema operativo Android emplea SQLite como sistema de gestión de bases de datos relacionales.

Algunas de las principales características de SQLite son:

- La biblioteca se enlaza con el programa, pasando a ser parte integral del mismo.

- El programa empleará la funcionalidad de SQLite por medio de llamadas simples a subrutinas y funciones, lo que reduce la latencia en el acceso a la BD.
- Las definiciones, tablas, índices y datos de la BD se guardan como un único fichero estándar en la máquina host (el terminal móvil).

6.2.1 Elección del Manejo de la BD Local para los Recursos

La clase propia de Android que maneja las funciones de BD es android.database.sqlite. Incluye todas las clases de gestión de base de datos necesarias para que la aplicación controle su propia base de datos privada.

Para facilitar la tarea de la gestión de la base de datos local y comprobar cómo se realiza la integración de librerías de terceras personas en un proyecto Android, se ha decidido emplear la librería "**TableDB**" [31], desarrollada y distribuida gratuitamente por Javier Pérez Pacheco.

Esta librería ofrece al usuario la posibilidad de manejar de una manera sencilla su base de datos en Android, ya que permite crear un fichero XML simple conteniendo la estructura de las tablas que se quieran crear. Será la propia librería la que lea esos ficheros durante el arranque de la aplicación (o cuando el usuario lo estime oportuno) y cree las distintas tablas de la base de datos local (si no existen ya). Además, ofrece una API a través de la cual se pueden crear, modificar y eliminar registros de las tablas eficazmente.

6.2.1.1 Ficheros de Creación de Tablas

Los ficheros necesarios para que la librería TableDB genere las tablas de nuestra base de datos son los siguientes:

- **tablas.xml:** contiene la estructura de las tablas
- **valores_iniciales.xml:** contiene los registros con los que se quiere inicializar las tablas la primera vez que se carguen. (No se requieren datos iniciales para el desarrollo de este proyecto).

[31] http://android.javielinux.com/tabledb.php

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<database name="geobuddies" version="2">
    <table name="recursos" to-string="%nombre%">
        <field name="modificado" obligatory="true" type="text" default="true" />
        <field name="nombre" obligatory="true" type="text" size="128" />
        <field name="descripcion" obligatory="false" type="text" />
        <field name="etiquetas" obligatory="false" type="text" />
        <field name="comentario" obligatory="false" type="text" />
        <field name="latitud" obligatory="true" type="text" size="128" default="0.0" />
        <field name="longitud" obligatory="true" type="text" size="128" default="0.0" />
        <field name="imagen" obligatory="false" type="text"/>
        <field name="usuario_id" obligatory="true" type="text" />
        <field name="path_ruta" obligatory="false" type="text" />
    </table>
    <table name="valoraciones" to-string="%valoracion%">
        <field name="valoracion" obligatory="true" type="integer" default="0" />
        <field name="recurso_id" type="foreign-key" foreign-table="recursos" />
    </table>
</database>
```

Ilustración 27 - tablas.xml

6.2.1.2 Integración de la librería TableDB en el proyecto

Una vez asociada la librería al proyecto, el uso es muy sencillo. Al estar basada en el patrón "Singleton", mediante el cual se tiene una única instancia de la clase principal, al arrancar la aplicación se genera la base de datos pasando como parámetros los ficheros ya vistos.

Para la aplicación en desarrollo, se generará dicha instancia una vez validada la sesión de usuario (en la clase FormLogin.java). Al no requerir valores iniciales, dicho fichero se creará vacío.

```
public class FormLogin extends Activity {

    • • •

    private TableDB tabla_bd;

    • • •

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.login);

        this.mContext = this;

        // Conexión con la librería TableDB
        try {
            TableDB.getInstance().open(this, R.xml.tablas, R.xml.valores_iniciales);
        } catch (Exception e) {
            e.printStackTrace();
        }
        tabla_bd = TableDB.getInstance();
     • • •
```

Ilustración 28 - Instanciación de la Librería TableDB

Este código establecerá la conexión con la librería, quien creará la base de datos si ésta no se ha creado en una ejecución anterior.

Por otro lado, cuando se quiere insertar un registro, como por ejemplo cuando el usuario agregue un recurso nuevo, se procederá de la siguiente manera:

```
// Guardamos en la BD
// Primero se crea la entidad recurso
Entity recurso = new Entity(DATABASE_TABLE);
recurso.addAllAttributes();
recurso.setValue("nombre", nombreRecurso);
recurso.setValue("descripcion", descrRecurso);
recurso.setValue("etiquetas", etiquetasRecurso);
recurso.setValue("comentario", comentRecurso);
recurso.setValue("latitud", lat);
recurso.setValue("longitud", lon);
recurso.setValue("imagen", pathImagen);
recurso.setValue("path_ruta", pathRuta);
recurso.setValue("usuario_id", user_id);
// Se marca el recurso como modificado por ser nuevo
recurso.setValue("modificado", "true");

// Luego se guarda en la tabla de la BD
long recursoId = recurso.nextId();
boolean i = recurso.save();
```

Ilustración 29 - Creación de un Registro en la BD

El lector podrá apreciar que se debe crear una instancia de la clase Entity, que representa un registro de la tabla pasada como parámetro. Acto seguido se deben añadir a la entidad todos los atributos, cuyos valores se irán asignando según sea necesario. Por último, para hacer efectivo el cambio en la base de datos, se deberá invocar el método "save" del objeto "Entity".

6.2.2 Estructura de la Base de Datos

Lo único que interesará mantener localmente será los recursos del usuario y la valoración que éste le asigne. Por esta razón, se crean dos tablas asociadas de la siguiente manera:

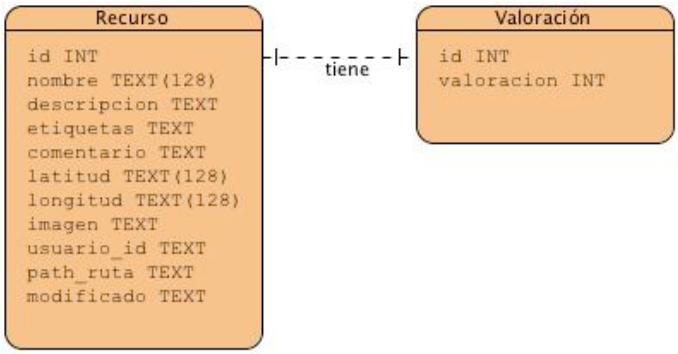


Ilustración 30 - Modelo ER de la Base de Datos Local

Nótese que no será necesario mantener una relación de usuarios de la aplicación, sino que sólo se deberá indicar, mediante un atributo, el identificador del usuario al que hace referencia el recurso en cuestión por si se diera el caso de que más de un usuario distinto utilizase la aplicación desde el mismo dispositivo.

Como identificador de usuario se empleará el nombre de inicio de sesión facilitado, salvo que se trate de un acceso anónimo a la aplicación, en cuyo caso se registrará el recurso bajo el identificador "anonimo".

6.2.2.1 Tabla "Recurso"

Campo	Tipo
id	INT
nombre	TEXT(128)
descripcion	TEXT
etiquetas	TEXT
comentario	TEXT
latitud	TEXT(128)
longitud	TEXT(128)
imagen	TEXT
usuario_id	TEXT
path_ruta	TEXT
modificado	TEXT

Tabla 2 – Estructura de tabla Recurso

Atributos:

- id: identificador del recurso.
- nombre: nombre del recurso.
- descripcion: descripción del recurso.
- etiquetas: cadena de texto que representa las etiquetas que asigna el usuario al recurso al darlo de alta.
- comentario: comentario personal sobre el recurso.
- latitud y longitud: ubicación geográfica del recurso.
- imagen: cadena de texto que representa la ubicación de la imagen.
- usuario_id: usuario que ha dado de alta el recurso.
- path_ruta: cadena de texto que representa la ubicación del archivo *.klm de ruta.

- modificado: representa si el archivo ha sido modificado y, por tanto sincronizado, o no (**true** -> no sincronizado; **false** -> sincronizado).

6.2.2.2 Tabla "Valoracion"

Campo	Tipo
id	INT
valoracion	INT

Tabla 3 – Estructura de tabla Valoracion

Atributos:

- id: identificador de la valoración.
- valoracion: valoración numérica del recurso (del 1 al 5).

6.3 Modelo de Datos Remoto

En cuanto al modelo de datos remoto, encargado de albergar los datos persistentes del sistema, no es propio de la aplicación, sino que depende de los servicios Web con los que se comunicará y que fueron ya implementados en un proyecto anterior de GeoBuddies.

La comunicación entre aplicación y servicios Web se realiza mediante el protocolo SOAP, como ya se explicó en el apartado 6 de esta memoria. A dichos servicios se accede a través de los ficheros WSDL (Web Services Description Language) alojados en las siguientes URLs:

1. Servicios relacionados con los usuarios: http://castor.dia.fi.upm.es:21917/GeoBuddies/UsuariosService?WSDL
2. Servicios relacionados con los recursos: http://castor.dia.fi.upm.es:21917/GeoBuddies/RecursosService?WSDL

Si se accede a las URLs en las que están alojados los esquemas, se podrá tener una idea de cómo está estructurada la base de datos remota:

3. Esquema de usuarios:
   http://castor.dia.fi.upm.es:21917/GeoBuddies/UsuariosService?xsd=1
4. Esquema de recursos:
   http://castor.dia.fi.upm.es:21917/GeoBuddies/RecursosService?xsd=1

A continuación se incluye la estructura de las entidades empleadas en el proyecto:

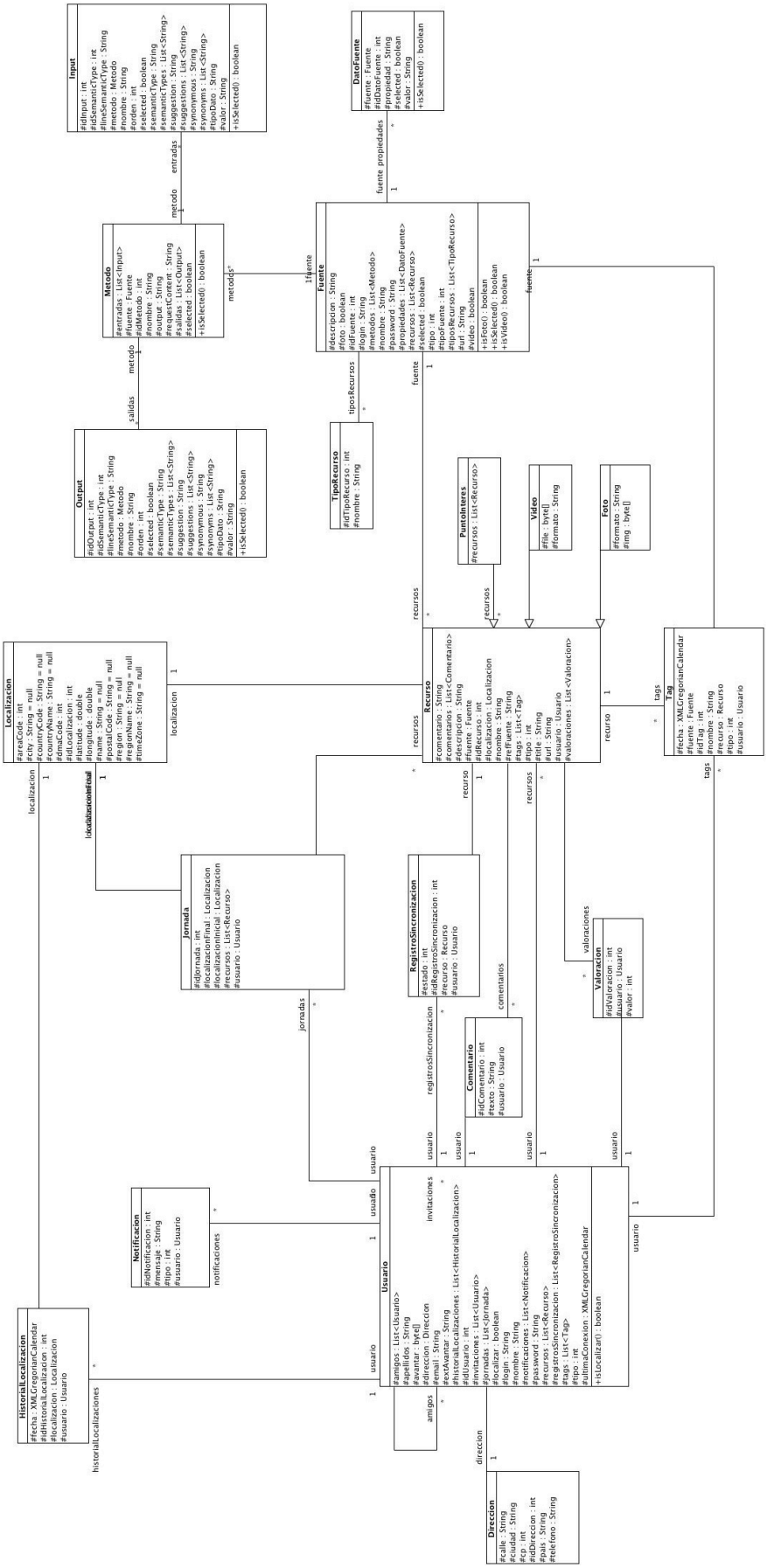


Ilustración 31 - Diagrama de clases de las estructuras del modelo de datos remoto

7 Implementación

A lo largo de este capítulo se pretende comentar los pasos seguidos para satisfacer los objetivos propuestos para este proyecto de fin de carrera:

1. Instalación del IDE de Eclipse 3.5 (Galileo).
2. Instalación de las herramientas de desarrollo (paquete ADT) sobre Eclipse: este plugin requerirá actualizaciones constantes, debido a las incesantes actualizaciones del sistema operativo.
3. Creación de un proyecto Android.
4. Creación de un AVD (configuración de dispositivo virtual para el emulador).
5. Ejecución sobre un emulador.
6. Firma de la aplicación.
7. Instalación sobre un dispositivo real.

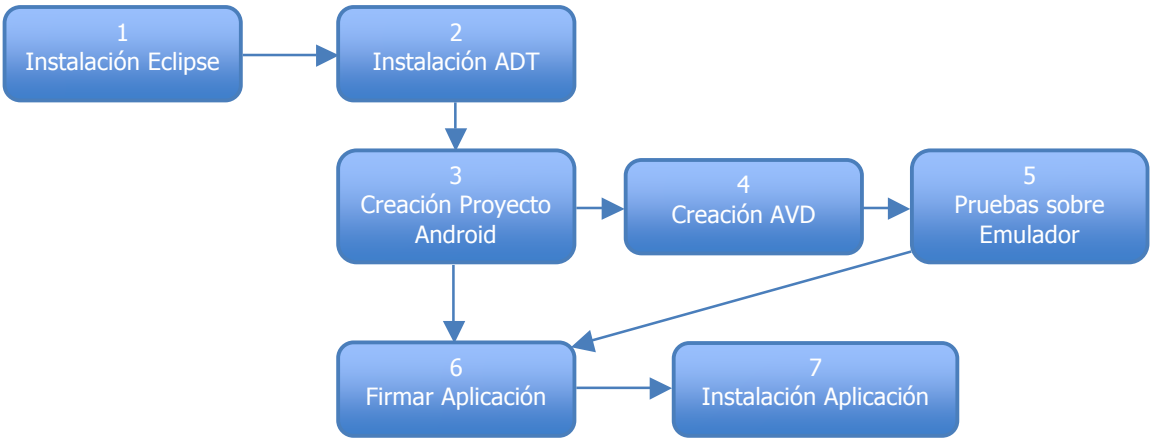


Ilustración 32 - Flujo de Tareas

7.1 Instalación del IDE: Eclipse 3.5 (Galileo)

Para el desarrollo de este proyecto se ha empleado el entorno de desarrollo Eclipse 3.5[32] (última versión disponible en el momento de implementación), ya que facilita la integración y el uso de las herramientas de desarrollo de Android.

Para instalar este IDE se seguirán los pasos que se especifican en el ejecutable descargado.

7.2 Instalación del plugin ADT sobre Eclipse

Tras la instalación del entorno de desarrollo, para poder implementar una aplicación para el sistema operativo Android, es necesario instalar el plugin de herramientas que facilita la creación de estas aplicaciones.

7.2.1 Preparación del equipo de desarrollo

Además del IDE recomendado, antes de proceder a la instalación del plugin de herramientas, es necesario instalar el paquete de desarrollo software de Android (SDK), para lo cual se debe instalar una versión compatible de las "SDK Tools" y, al menos, una plataforma de desarrollo.

La SDK se puede descargar de http://developer.android.com/sdk/index.html y consiste en un archivo que deberá descomprimirse en una ubicación segura del ordenador. Por defecto, los ficheros que componen la SDK se descomprimen en un directorio llamado "android-sdk-<plataforma>". Se hará referencia a este directorio durante la instalación de la ADT.

[32] Se ha procedido a instalar el "Eclipse IDE for Java EE Developers", que contiene la extensión para el desarrollo de aplicaciones Web.
http://www.eclipse.org/downloads/

7.2.2 Descarga del Plugin ADT

Se empleará el gestor de actualizaciones de Eclipse: Help -> Install New Software.

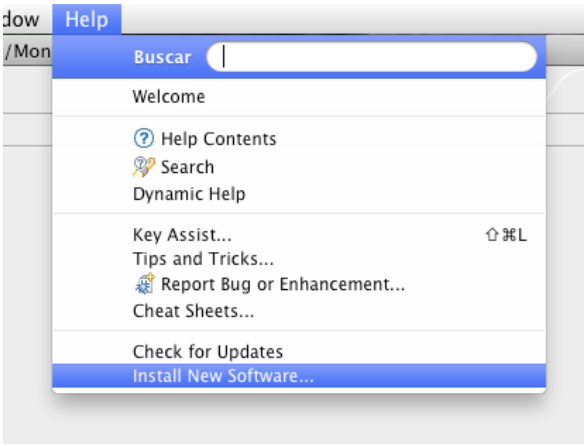


Ilustración 33 - Gestor de Actualizaciones de Eclipse

En la ventana de software disponible que aparece, pulsar sobre "Add" y agregar el siguiente sitio:

Name: Android Plugin
Location: `https://dl-ssl.google.com/android/eclipse/`

De vuelta a la pantalla anterior, se podrá apreciar que se ha agregado el sitio "Developer Tools". Seleccionar las funcionalidades que incluye y pulsar "Next".

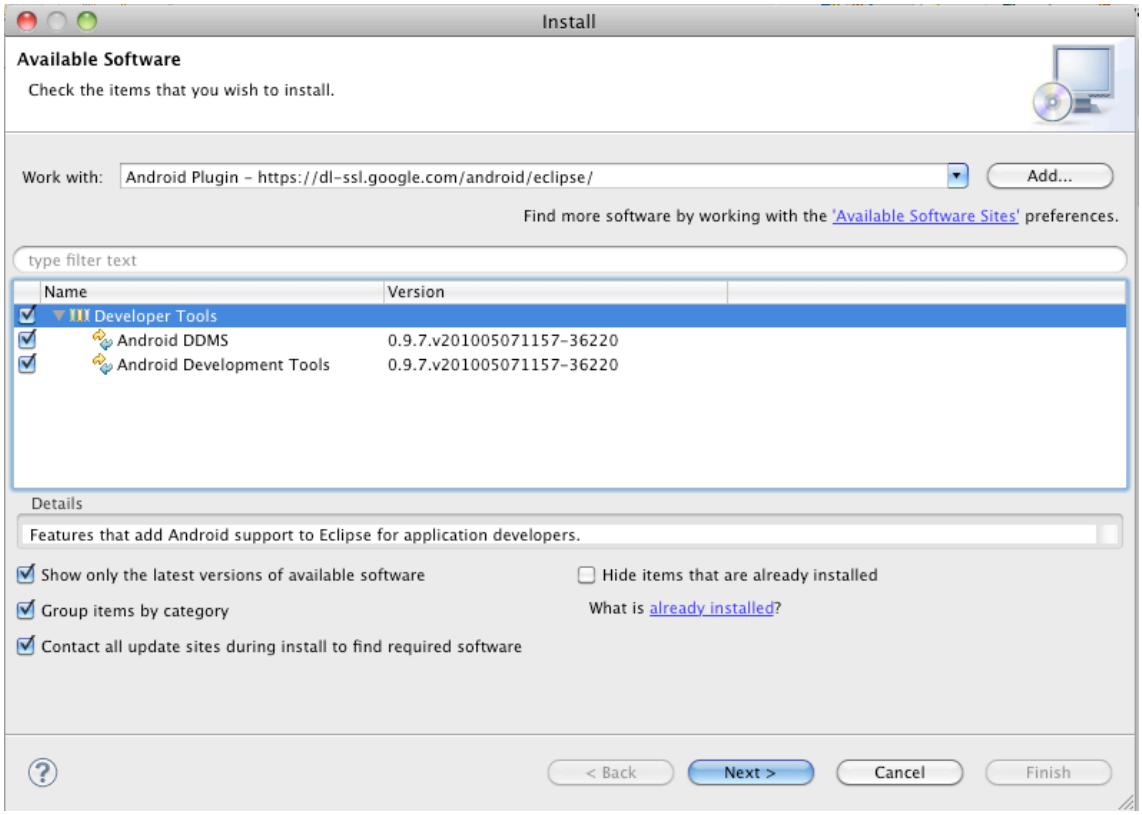


Ilustración 34 - Instalación ADT

Una vez calculadas las dependencias y leído el acuerdo de licencia, se pulsa de nuevo sobre "Next" y por último sobre "Finish" para completar la instalación.

7.2.3 Configuración del Plugin ADT

Antes de proseguir, se debe reiniciar Eclipse para que surjan efecto todos los cambios. Después será necesario configurar el plugin instalado para que apunte al directorio donde se descomprimieron los ficheros de la SDK.

Ilustración 35 - Acceso a las preferencias del IDE

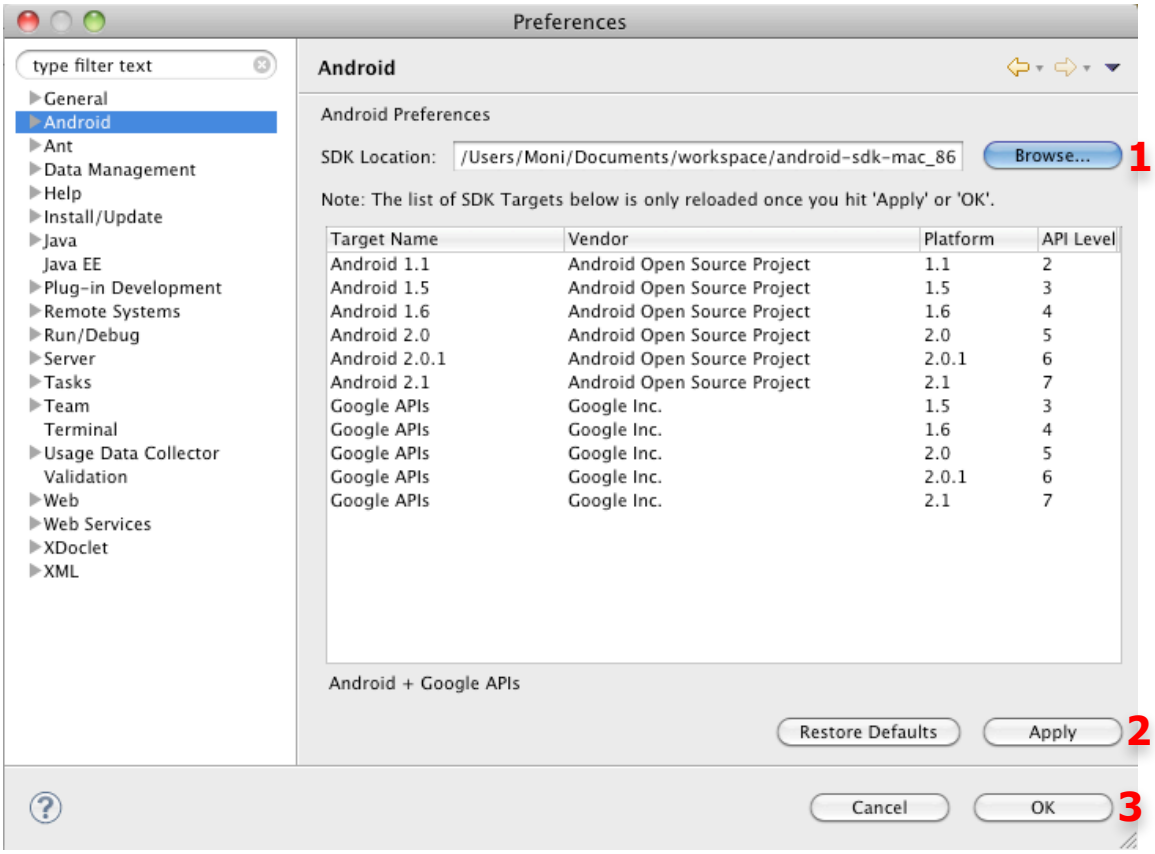


Ilustración 36 - Ubicación de los ficheros de la SDK

7.3 Creación de un Proyecto Android

El plugin recién instalado permite crear un proyecto Android de una manera muy sencilla.

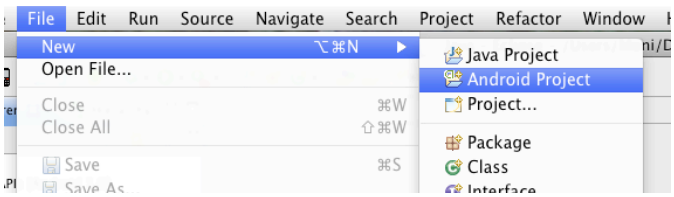


Ilustración 37 - Nuevo Proyecto Android

Posteriormente, se deben especificar los contenidos del proyecto:

- Nombre: Nombre de la carpeta donde se creará el proyecto.

- Contenido: Seleccionar "Create new Project in workspace" y elegir una ubicación para el directorio de trabajo.
- Plataforma objetivo: versión de la plataforma para la que se quiera desarrollar la aplicación.
- Propiedades: se deberá indicar el nombre de la aplicación (título que tendrá y que aparecerá en el dispositivo); nombre de paquete (espacio de nombres del paquete bajo el que residirá todo el código fuente); se seleccionará la opción "Create Activity" y se indicará el nombre de la actividad principal de la aplicación; y se introducirá la versión de SDK mínima necesaria para que la aplicación ejecute correctamente.

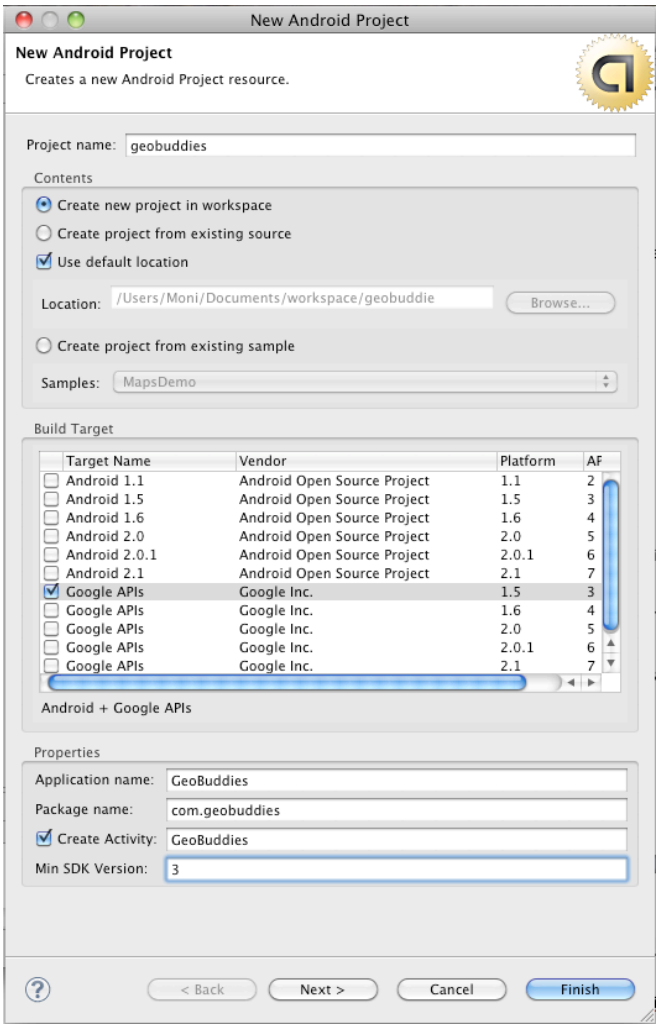


Ilustración 38 - Datos del nuevo proyecto

Una vez se haya completado el asistente de creación de proyecto nuevo, el ADT crea las siguientes carpetas y ficheros en el proyecto:

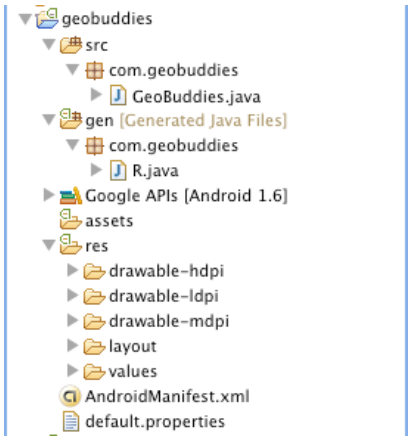


Ilustración 39 - Estructura del nuevo proyecto

- **src/** Incluye un esqueleto de la actividad principal.
- **gen/** Contiene los ficheros Java generados por el ADT.
- **Versión de Android** que tiene el *.jar contra el que se construirá la aplicación.
- **assets/** Carpeta vacía utilizable para almacenar distintos ficheros.
- **res/** Contiene los recursos de la aplicación: imágenes, ficheros de diseño de interfaz, valores de cadenas de texto, etc.
- **AndroidManifest.xml** Fichero con los permisos y actividades de la aplicación.
- **default.properties** Preferencias del proyecto.

7.4 Creación de un AVD

Un AVD es un dispositivo virtual de Android, con una configuración específica para el emulador, que permite modelizar dispositivos reales. Es la única manera de poder ejecutar una instancia del emulador, ya que necesita el fichero de configuración.

Eclipse permite crear un dispositivo virtual accediendo al panel de gestión de dispositivos virtuales:

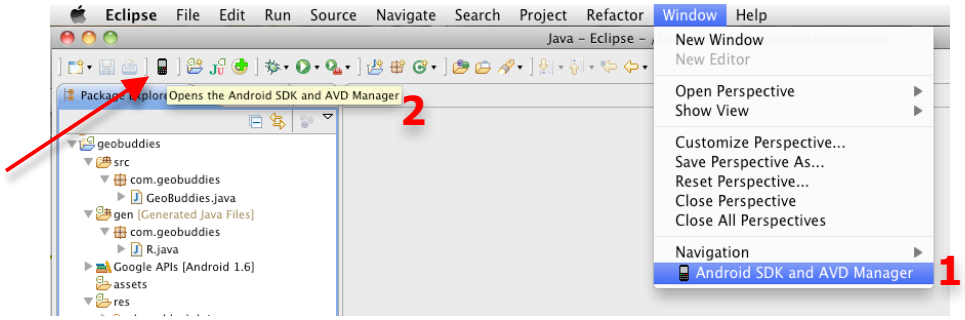


Ilustración 40 - Métodos de acceso al panel de gestión de AVDS

El panel de gestión de dispositivos virtuales muestra una lista de los dispositivos creados. Para crear uno nuevo, se pulsará sobre "New" y se rellenarán los datos con la configuración deseada.

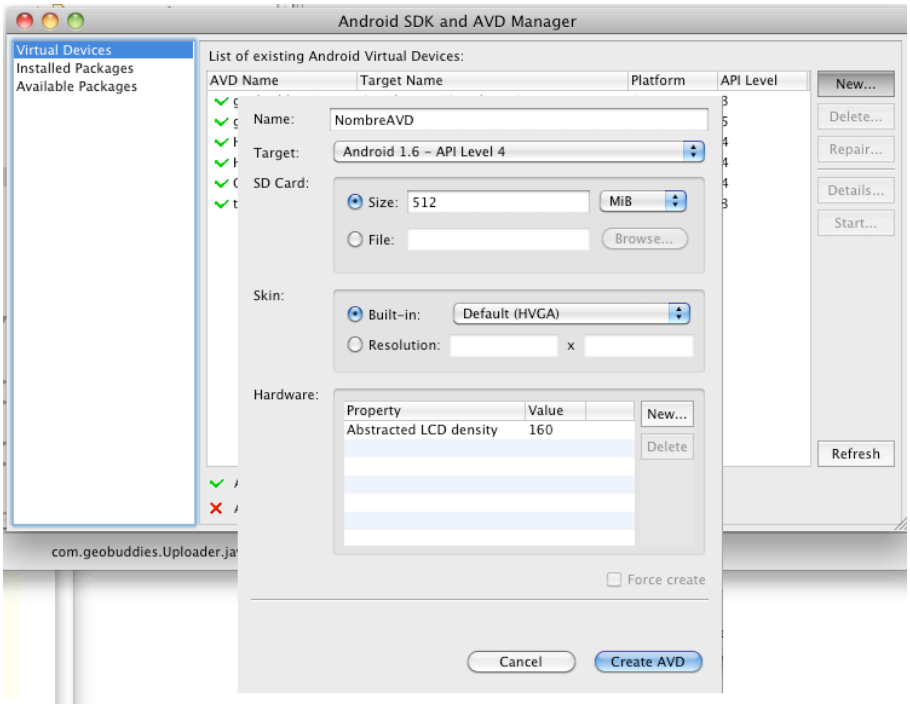


Ilustración 41 - Creación de un nuevos dispositivo virtual

7.5 Ejecución de la aplicación sobre un Emulador

Una de las ventajas de desarrollar una aplicación para Android es que proporciona un emulador, basado en un archivo de configuración, que facilita la prueba de compatibilidad del sistema en distintos dispositivos.

Una vez creado el AVD, o la configuración de un dispositivo virtual, se podrá ejecutar/depurar la aplicación sobre el mismo directamente desde el IDE de Eclipse.

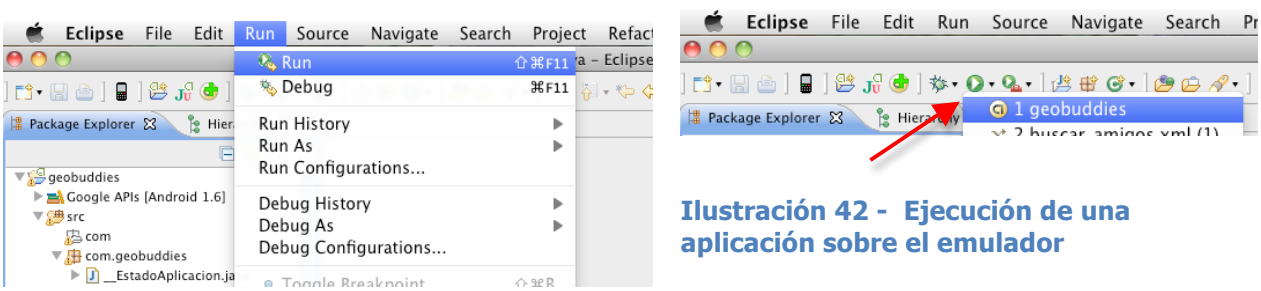


Ilustración 42 - Ejecución de una aplicación sobre el emulador

Al ejecutar la aplicación desde el menú "Run", el plugin ADT creará, automáticamente, una configuración de ejecución por defecto para el proyecto. Será entonces cuando el entorno de desarrollo llevará a cabo las siguientes tareas:

1. Compilación del proyecto, siempre y cuando haya habido cambios desde la última generación del ejecutable.
2. Si no existe una configuración por defecto para el proyecto, se creará una nueva (normalmente ocurrirá la primera vez que se ejecute el sistema).
3. Instala y ejecuta la aplicación en el emulador, que estará basado en el dispositivo destino definido en la configuración de ejecución.

7.5.1 Creación de la Configuración de Ejecución

Una configuración de ejecución especifica el proyecto que será ejecutado, la actividad que será iniciada, el emulador o el dispositivo conectado que se empleará, etc. Como se ha mencionado en el punto anterior, al ejecutar una aplicación Android por primera vez, el plugin ADT creará una configuración automáticamente. Esta configuración por defecto ejecutará la actividad por defecto del proyecto y usará el modo de selección de dispositivo destino automático (sin un AVD predefinido).

Sin embargo, gracias al panel de gestión de configuraciones de Eclipse, el desarrollador tiene la posibilidad de crear o modificar una configuración existente de manera que se adapte mejor a su aplicación.

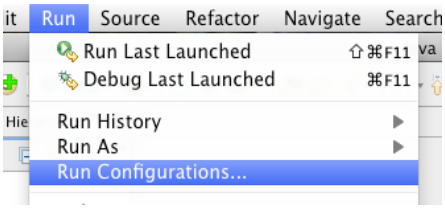


Ilustración 43 - Acceso al panel de gestión de configuraciones (AVDs)

A continuación se procede a mostrar la secuencia de acciones que se deben realizar para crear una nueva configuración. Nótese que, si lo que se desea es modificar una configuración ya existente, el usuario, únicamente, deberá seleccionarla del panel izquierdo y modificar aquellos datos que considere necesarios.

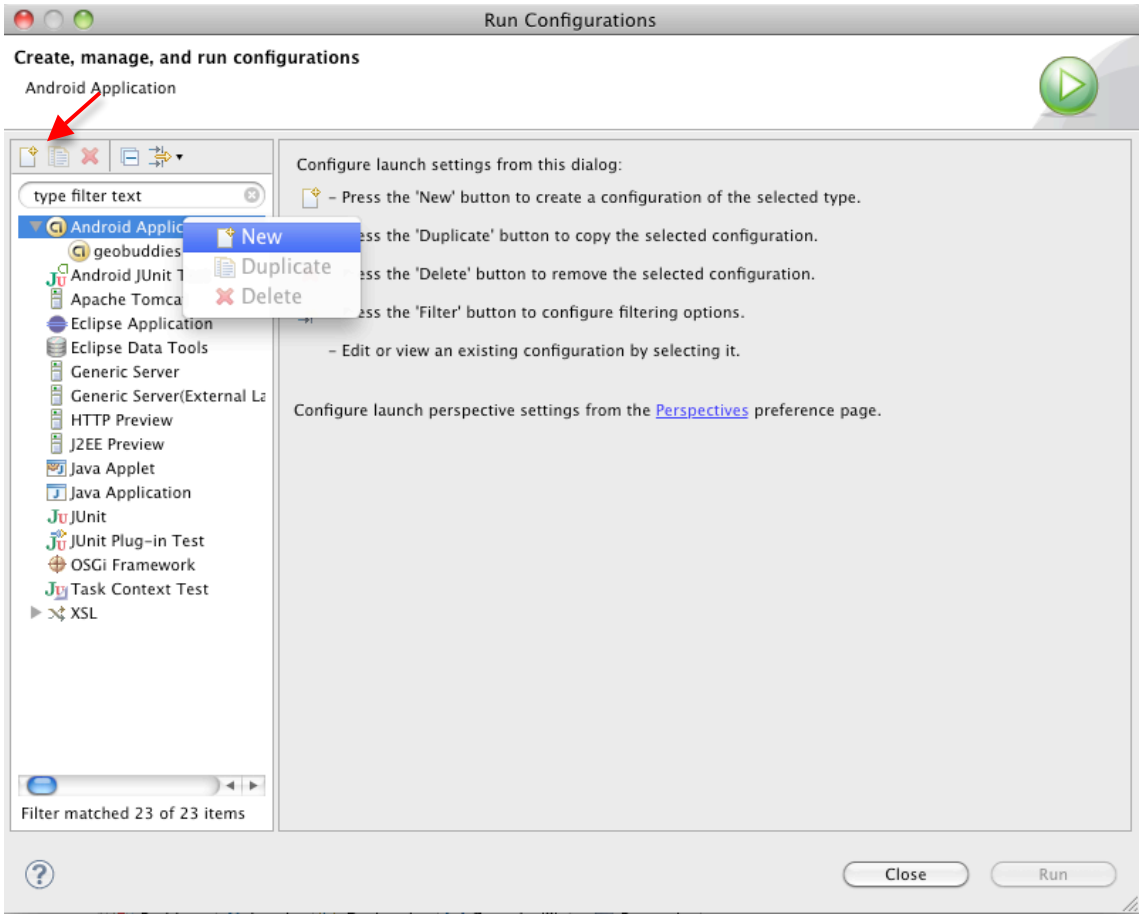


Ilustración 44 - Nueva configuración

Como se muestra en la figura anterior, existen dos formas rápidas de acceder a la pantalla de creación de un nuevo fichero de configuración de dispositivo virtual: o bien pulsando sobre el icono en el panel izquierdo, o bien desplegando el menú secundario sobre "Android Application".

Cuando el usuario haya accedido a la pantalla de creación de una configuración, deberá seleccionar el proyecto al que quiere asociar dicha configuración, la actividad que desea lanzar con la configuración, el tipo de dispositivo destino a emular (AVD asociado), así como opciones comunes de consola y disponibilidad en menús.

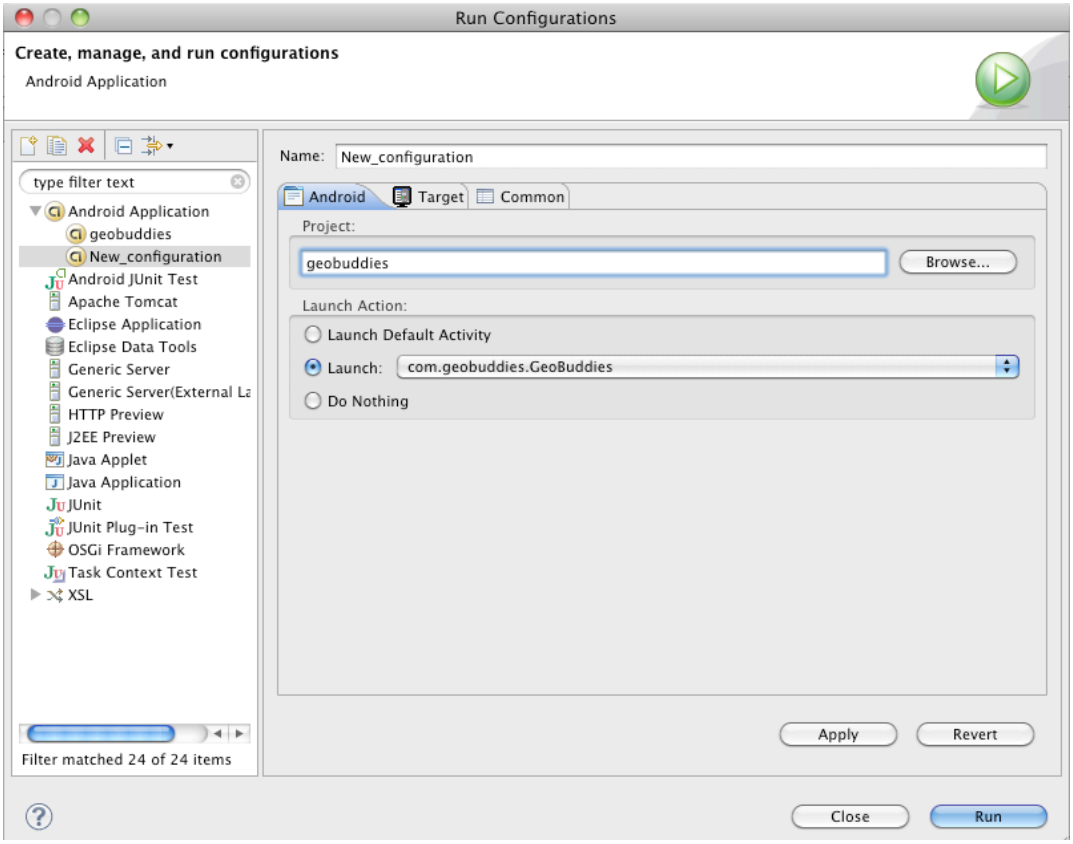


Ilustración 45 - Selección del proyecto y actividad

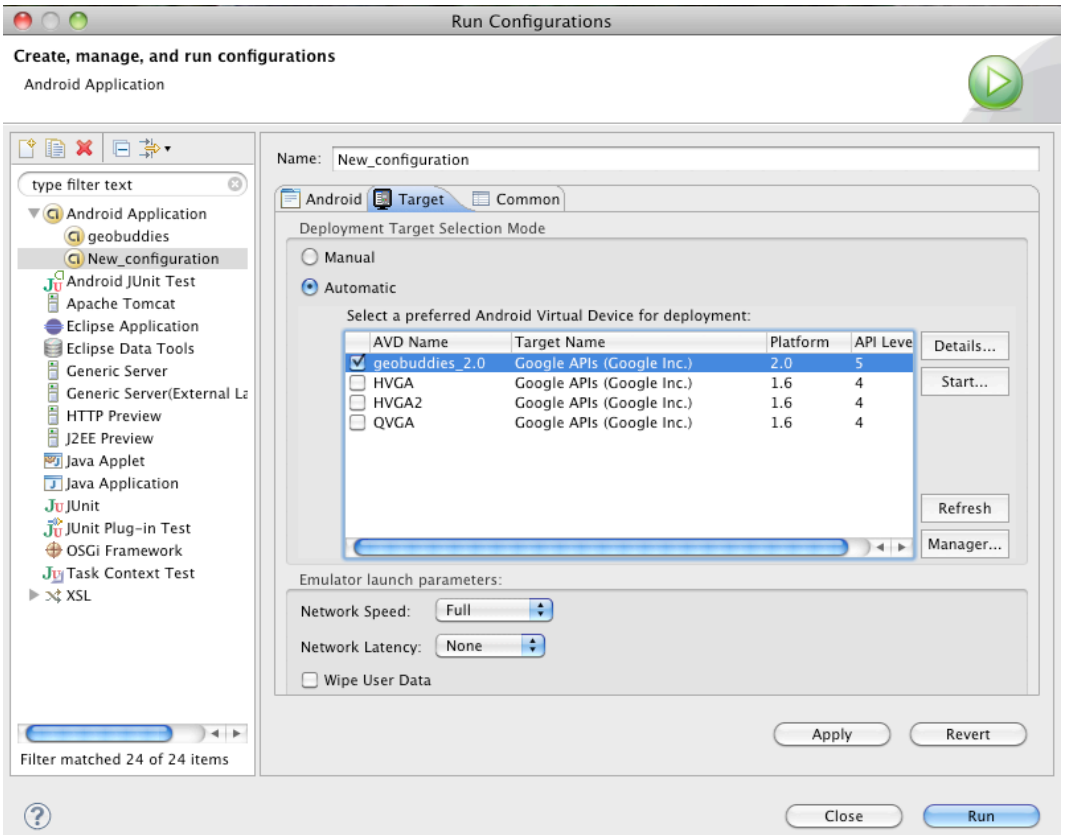


Ilustración 46 - Selección del AVD

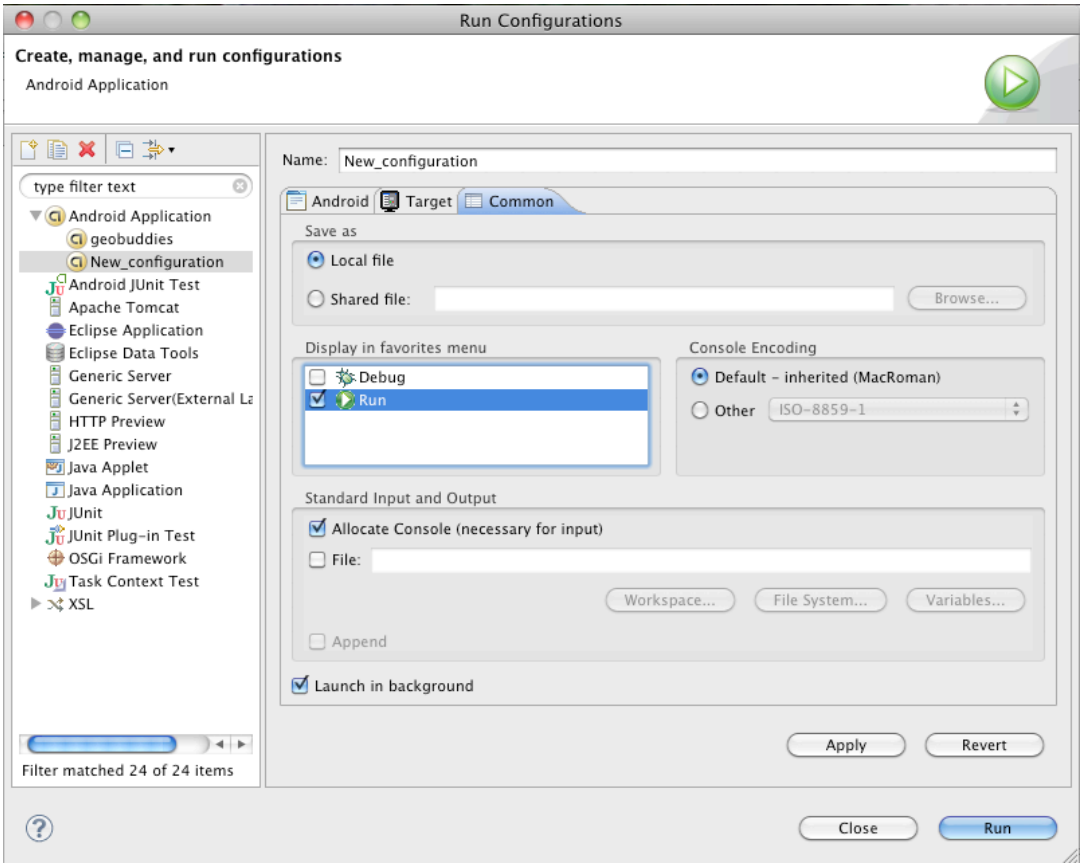


Ilustración 47 - Selección opciones comunes

En la pestaña de selección de dispositivo destino se puede elegir entre modo manual y automático.

7.5.2 Modo Manual y Automático para selección de AVD

Por defecto, una configuración de ejecución emplea el modo **automático** para seleccionar un dispositivo virtual. En este modo de funcionamiento, las utilidades de Android instaladas (ADT) elegirán un AVD para la aplicación de la siguiente forma:

1. Si ya hay un emulador en ejecución y su configuración se adecua a los requerimientos del ejecutable de la aplicación, ésta se instala y ejecuta sobre dicho emulador.
2. Si hay más de un emulador en ejecución que satisfaga los requerimientos de la aplicación, se muestra un selector de dispositivos para que el usuario pueda decidir sobre cuál de los dispositivos mostrados desea ejecutar la aplicación.

3. En caso de que no haya ningún emulador en ejecución, el ADT buscará entre los dispositivos virtuales una configuración adecuada para la aplicación. Si se encuentra un AVD que cumple esos requerimientos, se emplea ese dispositivo virtual para lanzar un nuevo emulador sobre el que instalar y ejecutar la aplicación.
4. En cualquier otro caso, no existe una configuración aceptable y la aplicación no podrá ejecutarse. Se mostrará un error por consola.

Por otro lado, si se ha seleccionado un AVD preferido ("Preferred AVD") durante la creación de la configuración de ejecución, la aplicación **siempre** se desplegará sobre ese AVD (si no hay un emulador ejecutando esa configuración, se lanzará uno nuevo).

Cuando una configuración de ejecución emplea el modo **manual**, se presentará al usuario un selector de dispositivos para seleccionar el dispositivo virtual sobre el que quiere ejecutar la aplicación.

7.6 Firma de la Aplicación

Toda aplicación desarrollada para Android debe ser firmada digitalmente por su creador antes de ser instalada en un dispositivo (o emulador) con dicho sistema operativo, ya que éste no permitirá la instalación de aplicaciones no identificadas.

Existen dos maneras de firmar una aplicación:

1. Mediante una clave de depuración, para realizar pruebas de manera inmediata. No se necesita realizar ninguna acción específica siempre que el plugin ADT tenga acceso a la herramienta de claves "Keytool".
2. Mediante una clave privada, necesaria para distribuir la aplicación.

El sistema operativo Android emplea el certificado de la firma digital de la aplicación como medio para identificar al autor de la misma y establecer relaciones de confianza entre aplicaciones. Tal y como se vio en el estado del arte de los sistemas operativos móviles, Android no emplea el certificado para controlar qué tipo de aplicaciones puede o no instalarse un usuario.

Existen una serie de puntos importantes que conviene saber para entender cómo funciona la firma de las aplicaciones basadas en Android:

- El sistema no instalará ninguna aplicación que no haya sido firmada.
- No se requiere una autoridad de certificación, sino que es suficiente con emplear certificados firmados personalmente.
- Mientras la aplicación se encuentra en estado de desarrollo, es posible utilizar la clave de depuración que generan las herramientas de la SDK. Sin embargo, cuando la aplicación está lista para publicarse, se debe firmar con una clave privada.
- El sistema comprueba la fecha de expiración del certificado en el momento de instalar la aplicación. Por tanto, si el certificado expira una vez instalada ésta en el dispositivo, la aplicación seguirá funcionando correctamente.
- Con la SDK se incluyen las herramientas estándar necesarias para firmar aplicaciones: **Keytool** y **Jarsigner**. Se podrá emplear esta herramienta para firmar los ficheros *.apk.
- Una vez firmada la aplicación, el desarrollador puede utilizar la herramienta "zipalign" para optimizar el paquete *.apk final.

7.6.1 Firma en Modo Depuración

Las herramientas de generación de código de Android proporcionan un método de firma en modo depuración para facilitar el desarrollo y las pruebas de la aplicación. Cuando se genera el ejecutable en este modo, las herramientas de la SDK invocan la herramienta "Keytool" para crear, automáticamente, una clave y un espacio de claves de depuración. Esta clave se utilizará para firmar el *.apk.

Los datos de la clave de depuración son los siguientes:

- Nombre del espacio de claves: "debug.keystore"
- Contraseña del espacio de claves: "android"
- Alias de la clave: "androiddebugkey"
- Contraseña de la clave: "android"

Para permitir que el ADT pueda emplear el certificado de depuración por defecto cuando se ejecuta la aplicación sobre un emulador, el desarrollador debe asegurarse que dicho plugin tiene acceso a la herramienta "Keytool".

Se puede indicar o cambiar la ubicación de la clave a utilizar desde el panel de "preferencias" del Eclipse:

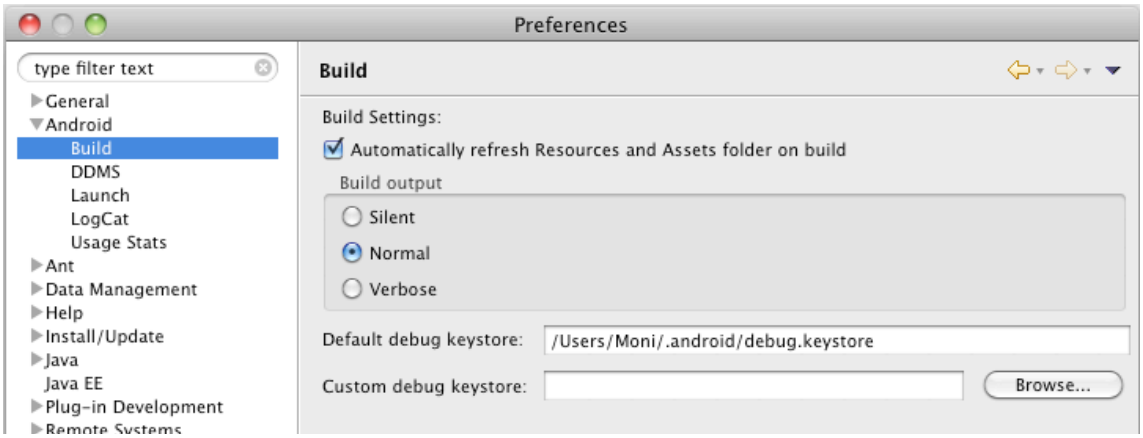


Ilustración 48 - Ubicación del fichero de claves

Al igual que cualquier otro certificado, el de depuración también tiene una caducidad. Desde que se genera por primera vez, el certificado tendrá una validez de 365 días, después de la cual se obtendrá un error de compilación en la consola de Android del IDE.

```
debug:
[echo] Packaging bin/samples-debug.apk, and signing it with a debug key...
[exec] Debug Certificate expired on 8/4/08 3:43 PM
```

Ilustración 49 - Error al expirar la firma de depuración

La forma de solucionar este problema es borrando el archivo "debug.keystore". De esta manera, la próxima vez que se genere el código de la aplicación, las herramientas de Android regenerarán un nuevo espacio de claves y una nueva clave de depuración.

7.6.2 Firma en Modo Publicación

Los pasos a seguir antes de poder distribuir la aplicación son los siguientes:

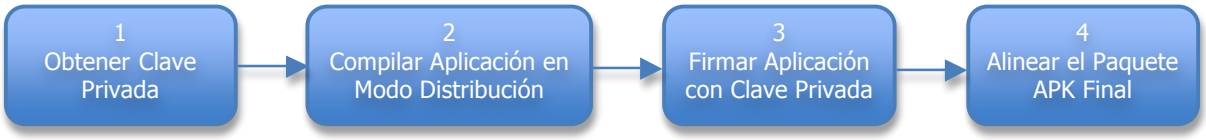
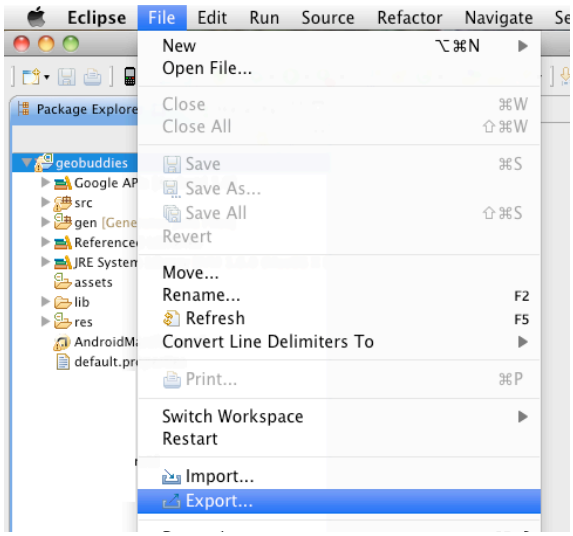


Ilustración 50 - Tareas para distribuir una aplicación Android

El entorno de desarrollo Eclipse tiene un ayudante de exportación que permite realizar los tres últimos pasos de una manera cómoda y sencilla. Incluso, en caso de ser necesario, permite obtener una nueva clave privada.

Este ayudante de exportación realiza toda la interacción con las herramientas Keytool y Jarsigner, lo que permite al desarrollador llevar a cabo la firma de la aplicación desde una interfaz gráfica en lugar de emplear los procedimientos manuales para compilar, firmar y alinear el paquete.



1. Primero se seleccionará el proyecto en el explorador de paquetes del IDE y se seleccionará **File -> Export...**

Ilustración 51 - Exportar Aplicación

2. A continuación, se seleccionará "Export Android Application", en la carpeta de Android, y se pulsará sobre "Next".

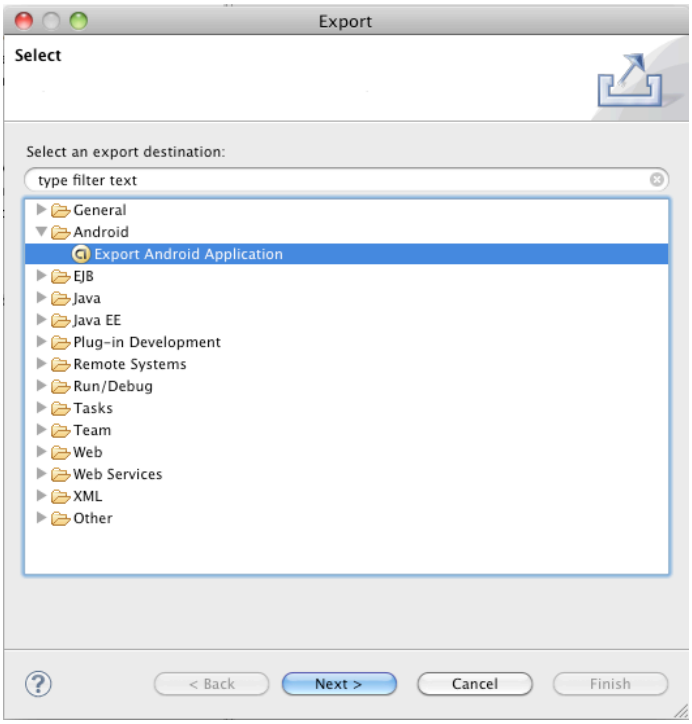


Ilustración 52 - Iniciar el proceso de exportación

3. A partir de este momento comienza el proceso de exportación, a través del cual el desarrollador será guiado por el asistente. El proceso incluye los pasos para la selección de la clave privada con la que firmar el *.apk.

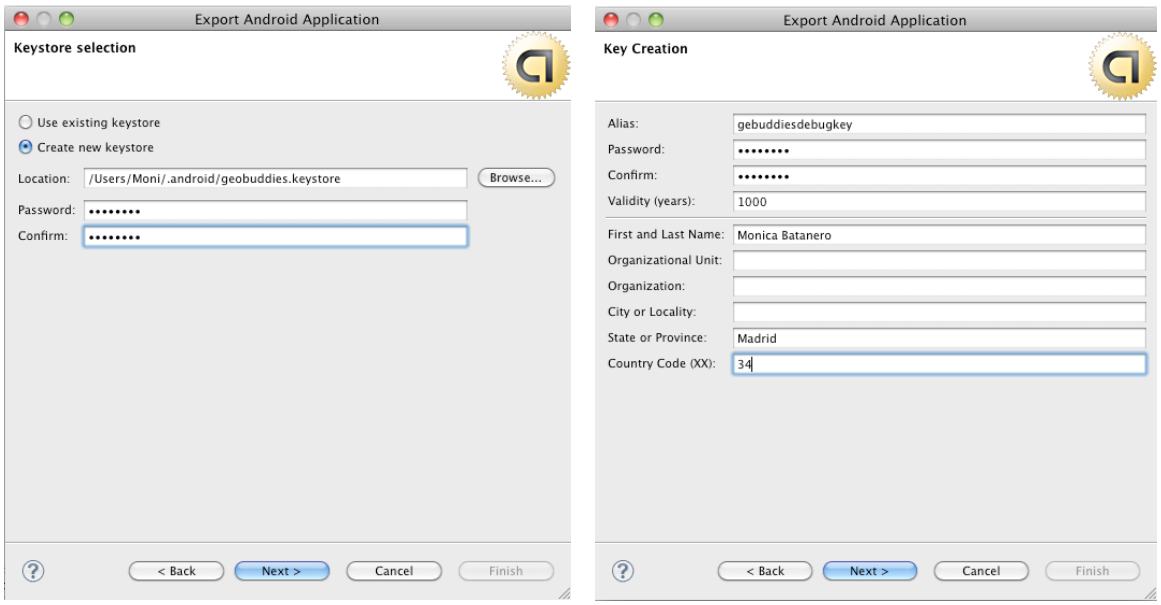


Ilustración 53 - Creación de la clave privada

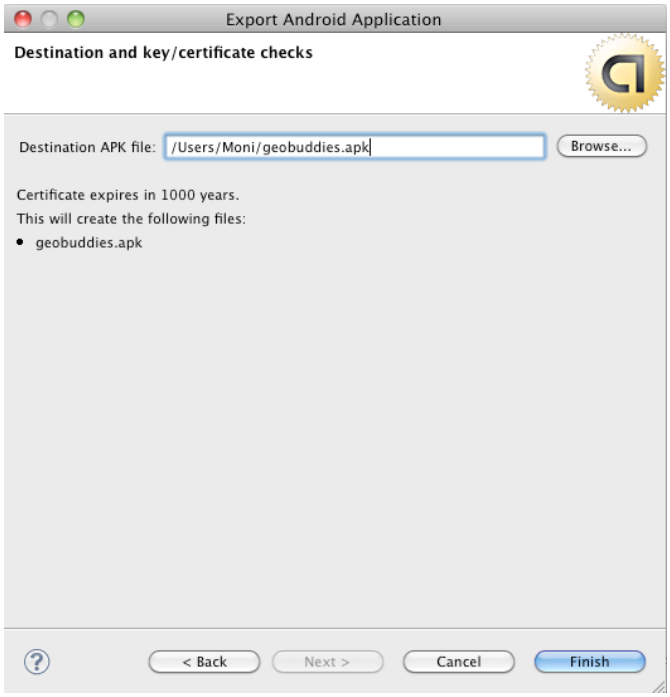


Ilustración 54 - Ubicación del paquete APK

7.7 Instalación de la Aplicación sobre un Dispositivo Real

Finalmente, una vez se posee el paquete de la aplicación Android (APK), el desarrollador podrá incluirlo en el Market (mercado de aplicaciones Android) o distribuirlo a través de su página Web personal, ya que lo único que necesita el usuario final es dicho archivo para instalarlo en su terminal.

Cualquier desarrollador que desee incluir su aplicación en el Android Market debe seguir estos tres pasos:

1. Crearse una cuenta de desarrollador en la URL siguiente: http://market.android.com/publish/signup
2. Pagar una cuota de alta de $25.00
3. Aceptar los términos de distribución de aplicaciones de Android: http://www.android.com/us/developer-distribution-agreement.html

Como ya se ha mencionado, si el desarrollador no desea acogerse a las ventajas que pueda suponer distribuir su aplicación en el Market (mayor impacto debido al gran número de usuarios de Android que buscan aquí nuevas aplicaciones), tiene la posibilidad de distribuir la aplicación por su cuenta siempre y cuando ésta esté firmada.

8 Evaluación del Sistema

En esta sección se describen las pruebas realizadas sobre la aplicación. En una primera parte se comentan las pruebas unitarias y de integración llevadas a cabo durante y después de la implementación, mientras que, en un segundo apartado, se comentarán las pruebas de usuario.

8.1 Pruebas Unitarias y de Integración

Para un correcto funcionamiento de la aplicación es de vital importancia que ésta se conecte satisfactoriamente con los servicios Web existentes. Así como para el resto de casos basta con ejecutar la aplicación y observar los resultados visuales, en este aspecto fue necesario observar que la estructura del mensaje enviado a través del protocolo SOAP era el adecuado, y que la información se recibía en la aplicación en la estructura y forma esperadas.

Para ello, se han implementado clases de prueba JUnit para las clases involucradas en la comunicación con dichos servicios: ServiciosRecursos.java y ServiciosUsuarios.java.

Para probar la integración del sistema se ha procedido a realizar las siguientes pruebas de interfaz.

8.1.1 Registro de un Nuevo Usuario

Al arrancar la aplicación, el usuario podrá elegir darse de alta en la comunidad virtual a través del siguiente enlace:

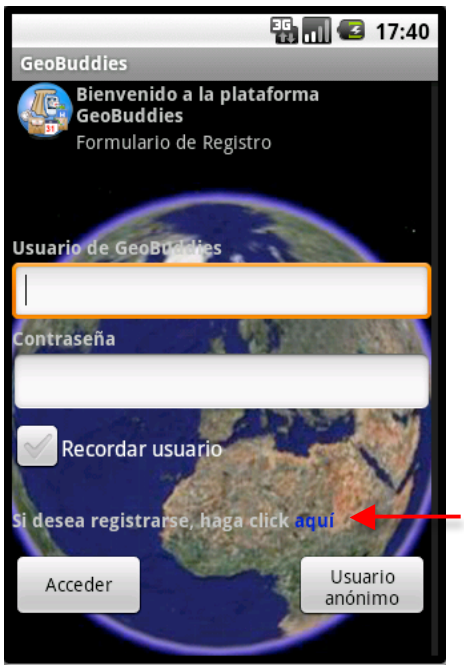


Ilustración 55 - Pantalla de inicio de sesión

A través de dicho enlace, el usuario accede al formulario de registro, que deberá cumplimentar para, después, pulsar "Aceptar".

Ilustración 56 - Formulario de registro

Como se puede apreciar, este panel permite verificar la disponibilidad del nombre de usuario, lo que admite cambiarlo antes de proceder con el registro.

Ilustración 57 - Comprobar disponibilidad del usuario (OK)

En este primer caso, el usuario puede comprobar que el nombre de usuario seleccionado (marcado en verde) no existe aún en la comunidad virtual, por lo que podría darse de alta con dicho nombre sin ningún problema.

En caso de que el nombre de usuario no esté disponible, se le mostrará al usuario en color rojo.

Si, por el contrario, el usuario decide no comprobarlo con anterioridad, se le mostrará un mensaje indicando que el nombre de usuario no está disponible (además de marcar el texto en rojo). Si el nombre de usuario fuese válido, el

sistema redirigiría al usuario a la pantalla de inicio rellenando el primer campo con el nombre de usuario escogido.

Ilustración 58 - Comprobar disponibilidad usuario (Error)

Ilustración 59 - Registro sin comprobación (Error)

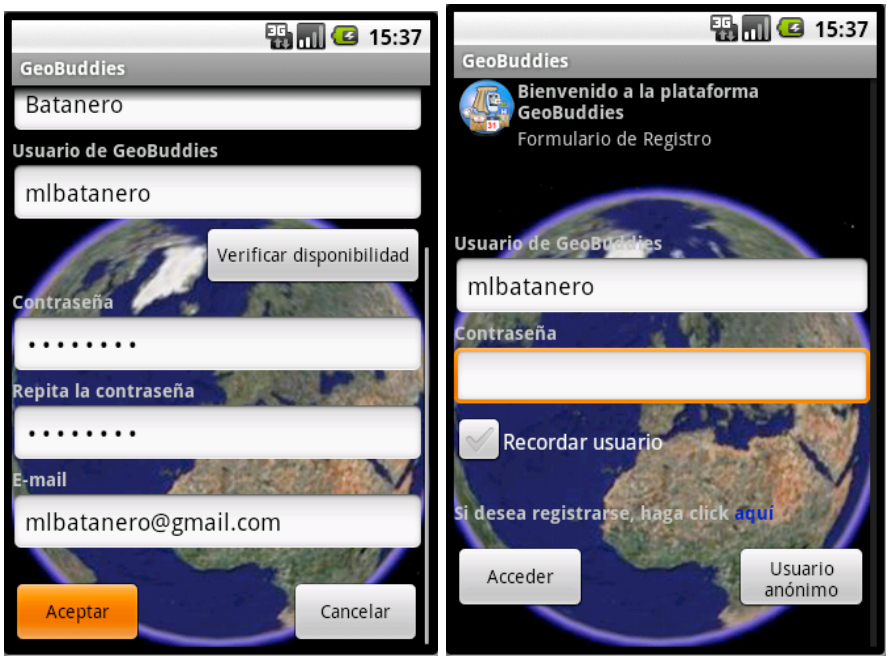


Ilustración 60 - Registro sin comprobación (OK)

8.1.2 Consulta de los Recursos del Usuario

Para poder consultar sus recursos, el usuario debe acceder a este panel desde el acceso directo de la pantalla principal. Una vez en "Mis Recursos", éstos se muestran en modo lista y con un pequeño icono acorde con la tipología del recurso.

Ilustración 61 - Acceso a "Mis Recursos"

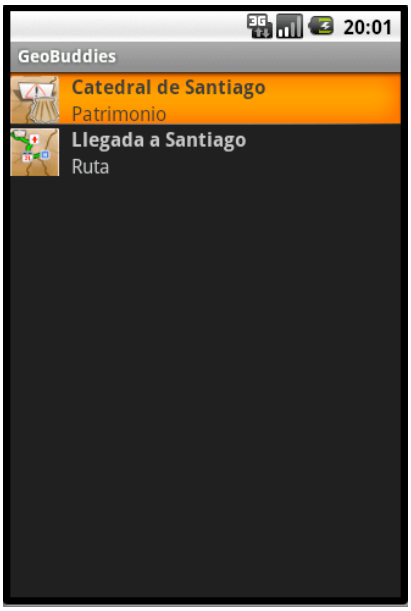
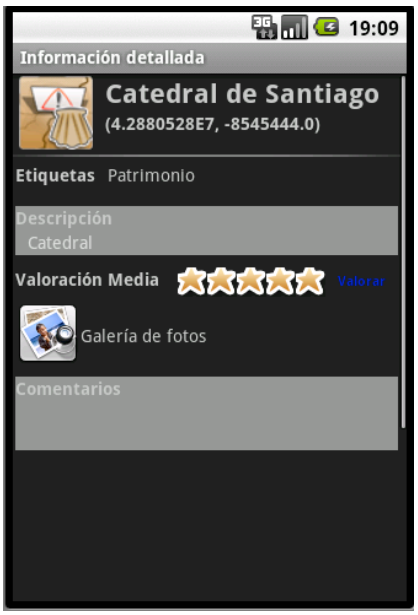


Ilustración 62 - Pantalla "Mis Recursos"

Cuando se carga la lista de resultados, el usuario puede proceder a consultar los detalles de uno en concreto pulsando sobre ese elemento.

8.1.2.1 Detalles de un recurso genérico



La vista detallada de un recurso general (cualquiera excepto una ruta) se compone de los siguientes campos:

- Nombre del recurso
- Coordenadas de ubicación geográfica
- Etiquetas con las que se registró
- Descripción
- Valoración otorgada por los usuarios
- Acceso a fotos (si las tiene)
- Comentarios de los usuarios

Ilustración 63 - Detalles de un recurso general

8.1.2.2 Detalles de un recurso de tipo "ruta"

La vista detallada de un recurso de tipo "ruta" está compuesta por los mismos campos mencionados anteriormente con la salvedad de que, además, incluye un botón de acceso a la visualización de dicha ruta sobre un mapa navegable (opciones de zoom y movimiento por el mapa).

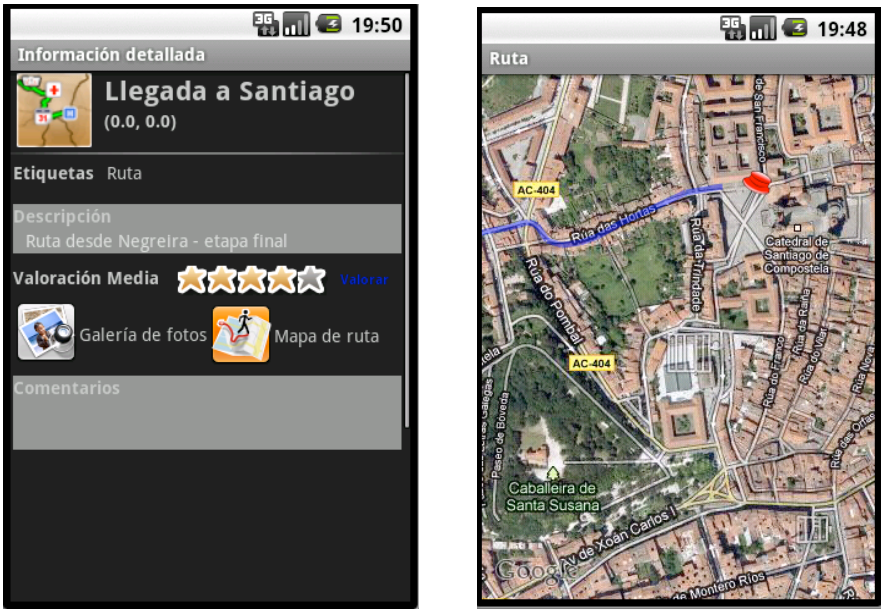


Ilustración 64 - Detalles y visualización de la ruta de un recurso

8.1.3 Consulta de los Amigos (y sus recursos) del Usuario

Para poder consultar sus contactos, el usuario debe acceder a este panel desde uno de los dos accesos directos de la pantalla principal. Una vez en "Mi Comunidad", éstos se muestran en modo lista.

Ilustración 65 - Acceso a "Mi Comunidad"

Al seleccionar un determinado amigo, se mostrarán sus recursos, que podrán verse en detalle del mismo modo que se ha comentado en la prueba anterior.

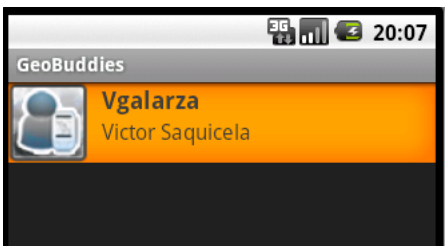


Ilustración 66 - Amigos del usuario

8.1.4 Agregar y Eliminar Amigos

Una vez en el panel de amigos del usuario, éste puede agregar nuevos contactos o eliminar algún contacto existente.

Si decide agregar un nuevo amigo, deberá pulsar sobre la tecla "Menú" del terminal para acceder al menú asociado a la pantalla.

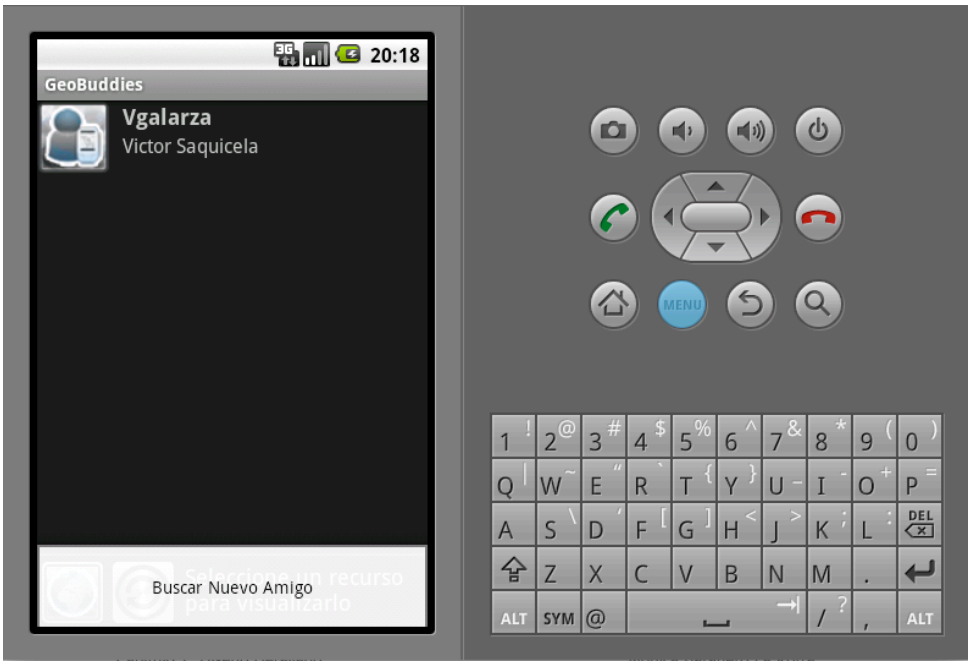


Ilustración 67 - Agregar Amigo

Se observa que se despliega un menú en la parte inferior de la pantalla. Si se pulsa sobre la opción "Buscar Nuevo Amigo", aparecerá una nueva pantalla en la que buscaremos un hipotético amigo llamado "Sebastián Saquicela" (ssaquicela) de tres maneras distintas:

Ilustración 68 - Búsqueda por nombre de usuario

Ilustración 69 - Búsqueda por nombre

Ilustración 70 - Búsqueda por apellidos

Incluso si el texto introducido no es exacto, la búsqueda devolverá los resultados más acordes a dicho texto.

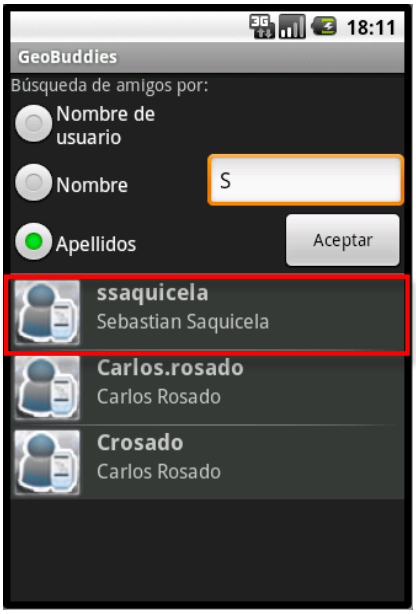


Ilustración 73 – Búsqueda aproximada

Ilustración 71 – Confirmación

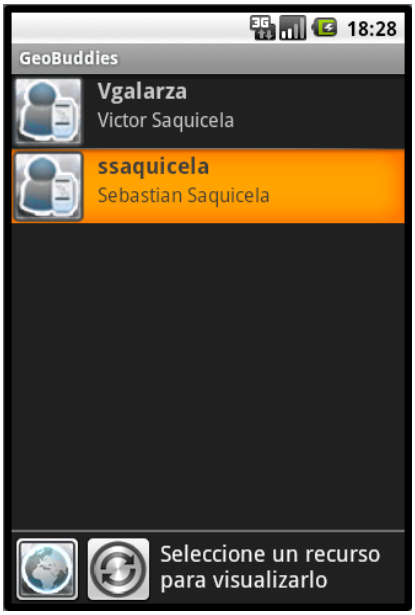


Ilustración 72 - Usuario agregado satisfactoriamente

Cuando la búsqueda ha sido satisfactoria, se puede proceder a agregar un amigo pulsando sobre él en la lista, lo que se confirmará añadiendo una nueva entrada en la lista que representa la comunidad del usuario.

En cuanto a la eliminación de un contacto de la lista, se deberá presionar durante un par de segundos sobre el usuario que deseamos eliminar. Pasado este tiempo, el sistema solicitará la elección de una acción del menú desplegable que aparece en pantalla. Si el usuario selecciona "Borrar Amigo", el sistema procederá con el borrado tras la solicitud de confirmación.

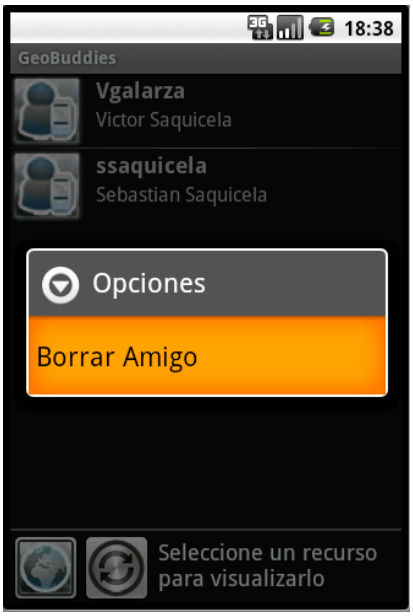


Ilustración 74 - Eliminar Amigo

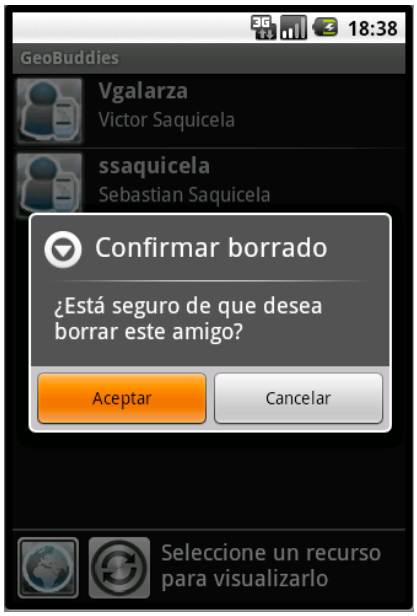


Ilustración 76 - Confirmación de borrado

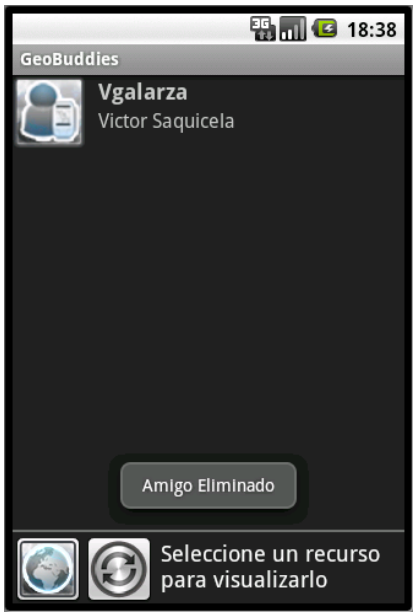


Ilustración 75 - Contacto eliminado

8.1.5 Registro de un Nuevo Recurso

Para agregar un nuevo recurso, el usuario puede proceder de varias maneras:

- A través del acceso directo situado en la barra de herramientas deslizante en la pantalla principal.
- Desde el mapa navegable.

Ilustración 77 - Modos de agregar nuevos recursos

En ambos casos se accede al formulario de recogida de datos del nuevo recurso. A continuación vemos el acceso desde el mapa por ser éste más elaborado.

En primer lugar, una vez sobre el mapa, el usuario se situará sobre aquella zona en la que desee agregar el nuevo recurso. Cuando el mapa esté centrado en la zona de interés, se deberá pulsar durante dos segundos aproximadamente sobre el punto en cuestión.

Por defecto, al acceder al mapa, éste se centra en la ubicación actual del usuario si es que puede obtenerse. En caso contrario, se centrará en Santiago de Compostela.

Al agregar un nuevo recurso, si se hace desde el mapa, se indicará la existencia de un recurso sobre el mapa mediante un icono flotante (que será acorde a la tipología del recurso). Dicho icono será un enlace a información más detallada del recurso.

Se recuerda que todos los recursos del usuario se pueden consultar desde el panel "Mis Recursos".

Ilustración 78 – Nuevo recurso sobre el mapa

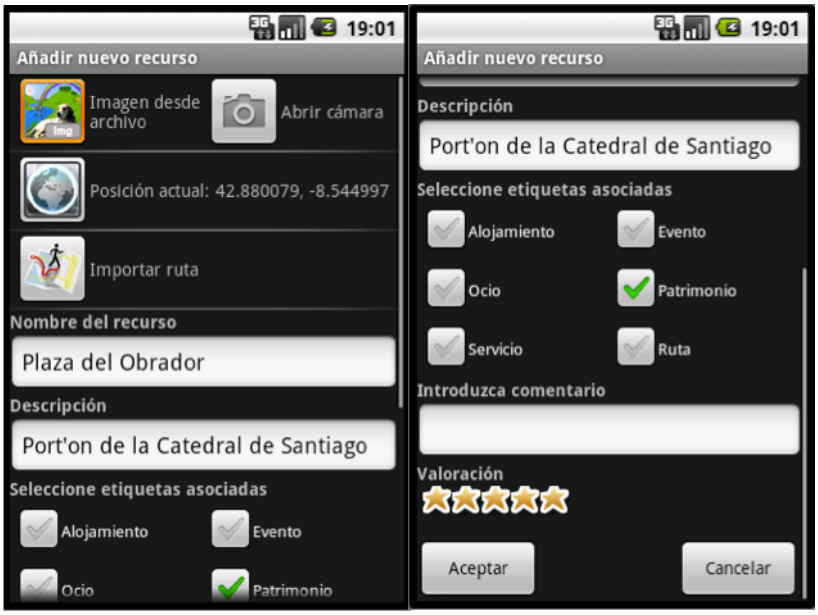
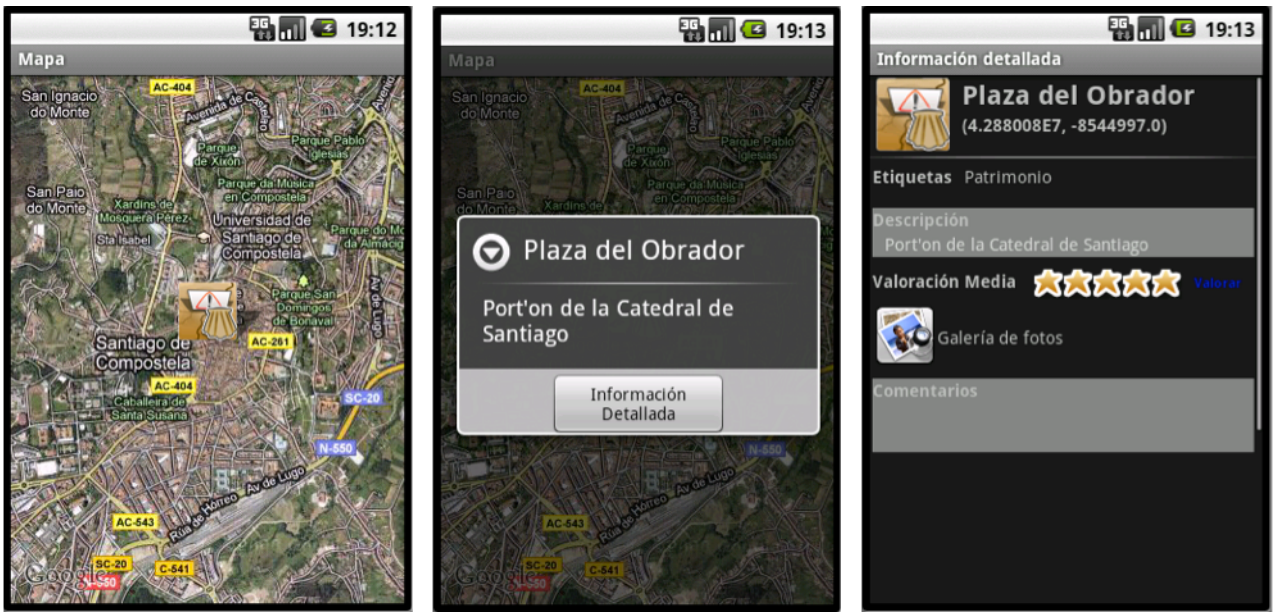


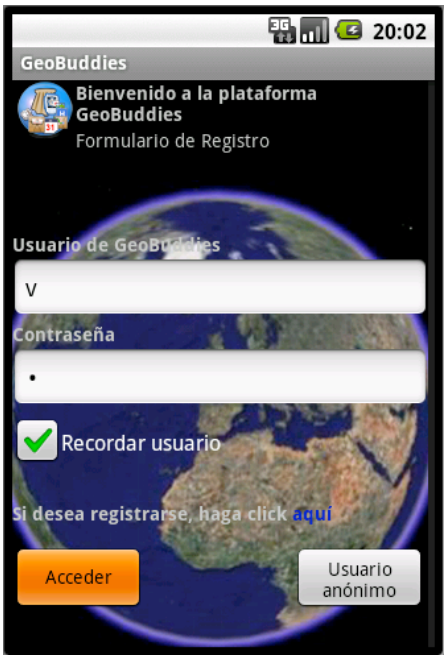
Ilustración 79 - Formulario para agregar un nuevo recurso

Resultado de agregar el recurso anterior sobre el mapa:



8.1.6 Inicio de Sesión y Panel de Preferencias

Cuando un usuario instala y arranca la aplicación por primera vez, se encuentra con una pantalla de inicio de sesión, en la que deberá registrarse (en caso de no haberlo hecho ya) para poder iniciar una sesión que le permita manejar sus recursos.



Se va a probar cómo funciona la opción "Recordar usuario" y cómo está ligada a las preferencias del sistema.

En la imagen de la izquierda se puede apreciar como el usuario "v" desea que su sesión se asocie al terminal para no tener que introducir sus datos cada vez que desee utilizar la aplicación.

En este caso, en sucesivas ejecuciones del sistema, el usuario accederá directamente a la pantalla principal.

Cuando se elige esta opción durante el inicio de sesión, se puede comprobar que, en las preferencias de la aplicación, la casilla de verificación "Activar recordar usuario" permanece activa. De no ser así, significa que la sesión no será recordada en posteriores ejecuciones, por lo que será aquí donde deba dirigirse el usuario para activar/desactivar esta funcionalidad.

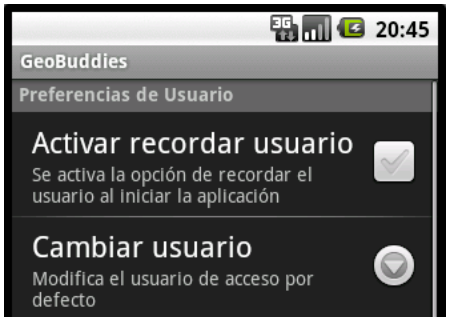


Ilustración 80 - Inicio de sesión no automático

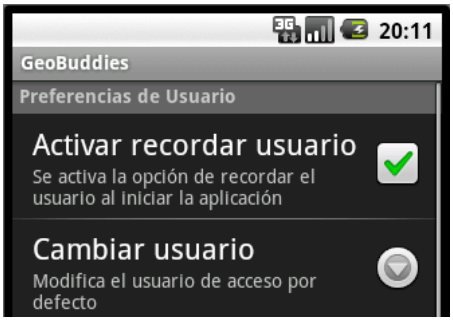


Ilustración 81 - Inicio de sesión automático

Se informa al lector de que esta funcionalidad se creó a partir de los comentarios proporcionados por los usuarios de prueba.

8.2 Pruebas de Usuario

Para la consecución de las pruebas de usuario, se han seleccionado candidatos atendiendo a dos criterios:

1. Usuarios familiarizados con el sistema operativo Android y, por tanto, con el tipo de aplicaciones que maneja.
2. Usuarios no familiarizados con el sistema operativo Android.

En ambos casos se ha procedido a introducir brevemente el objetivo del sistema y su finalidad, por lo que se ha pedido a los usuarios que se metiesen en el papel de un peregrino del Camino de Santiago para realizar las pruebas.

Los escenarios de familiarización intentan cubrir las funcionalidades principales implementadas en la aplicación para que todos los usuarios probasen, como mínimo, las mismas tareas y se puedan así obtener unas estadísticas fiables sobre cuestiones de interfaz de usuario y funcionalidad.

Las instrucciones, escenarios, encuesta de satisfacción, encuesta de impresiones y el formulario de anotaciones tipo empleado durante la recogida de datos en la evaluación de usabilidad del sistema se incluyen en el Anexo A.

A modo de información, antes de proceder con los resultados obtenidos de las pruebas de usuario, se quiere hacer constar que las pruebas han sido realizadas sobre un terminal HTC Magic, cuyas características de interés (con respecto a la aplicación) son las que siguen:

Procesador	Qualcomm® MSM7201a™, 528 MHz
Memoria	ROM: 512 MB RAM: 192 MB
Pantalla	Pantalla táctil TFT-LCD de 3,2 pulgadas con resolución HVGA (320 x 480)
Red	HSDPA/WCDMA: • 900/2100 MHz • Velocidad de subida de hasta 2 Mb/s y 7,2 Mb/s de bajada GSM/GPRS/EDGE cuatribanda: • 850/900/1800/1900 MHz (La banda de frecuencias y velocidad de datos dependen del operador)
Control del dispositivo	Trackball con botón Intro
GPS	Antena GPS interna

Conectividad	Bluetooth® 2.0 con Enhanced Data Rate y A2SP para auriculares inalámbricos estéreo Wi-Fi®: IEEE 802.11 b/g HTC ExtUSB™ (mini-USB 2.0 y conector de audio en uno)
Cámara	Cámara en color de 3,2 megapíxeles con enfoque automático
Formatos de audio compatibles	AAC, AAC+, AMR-NB, MP3, WMA, WAV, AAC-LC, MIDI y OGG
Formatos de vídeo compatibles	MP4 y 3GP
Ranura de ampliación	Tarjeta de memoria microSD™ (compatible con SD 2.0)
Características especiales	Acelerómetro Brújula digital

Tabla 4 - Características del dispositivo de pruebas (HTC Magic)

8.2.1 Resultados de la Evaluación por parte de los Usuarios

A partir de los formularios de pruebas obtenidos se han podido analizar los contenidos y la usabilidad del sistema. Seguidamente se muestran los comentarios de mejora de los usuarios, divididos según su tipología: Apariencia, Usabilidad y Funcionalidad.

Apariencia

1. Cambiar la imagen de fondo en la pantalla de inicio de sesión y en la pantalla principal de la aplicación por una más acorde con su temática.
2. Corregir el tamaño de las filas en las listas de resultados. En algunos casos son demasiado estrechas y se corta parte del contenido.
3. En general, se debería modernizar la interfaz.

Usabilidad

4. Cambiar la ubicación de los accesos a los paneles de "Búsqueda", "Añadir", "Comunidad", "Mapas", "Datos", "Ajustes" y "Salir" directamente a la pantalla principal.
5. Añadir iconos representativos en las listas de resultados de recursos para una identificación más rápida según el tipo de recurso.
6. Añadir avatar por usuario.
7. Añadir texto fijo en los botones inferiores de la pantalla "Comunidad".
8. Indicar en todo momento en qué pantalla se encuentra el usuario.

Funcionalidad

9. Añadir la opción "Cambiar Usuario" para permitir que un usuario distinto pueda acceder a la aplicación desde el mismo terminal.
10. Añadir la opción "Recordar Usuario" en el panel de inicio de sesión. Deberá tener su correspondiente "Activar Recordar Usuario" en el panel de "Ajustes" de la aplicación para su posterior desactivación.
11. Eliminar la opción "Selección de Mapas" del panel de "Ajustes", ya que la aplicación funciona únicamente con los mapas de Google, por lo que esta selección se hace innecesaria.

La siguiente tabla muestra, a modo de resumen, los puntos tratados y los que se han dejado para un desarrollo posterior de la aplicación.

# Problema	Descripción	Comentario	Resuelto
1	Imagen de fondo	Se ha establecido un fondo más acorde con la temática de la aplicación	
2	Tamaño listas resultados	Se amplía el alto de cada fila para que el texto se muestre correctamente	
3	Interfaz más moderna	Se deja para una implementación posterior por falta de tiempo en este TFC	
4	Ubicación accesos directos	Se ha creado una galería Android en la pantalla principal	
5	Iconos representativos recursos	Cada recurso lleva asociado un icono específico según su tipología	
6	Avatar usuarios	Se deja para una implementación posterior por falta de tiempo en este TFC	
7	Texto descripción botones "Comunidad"	Se incluyen descripciones junto a los botones del panel de usuarios	

# Problema	Descripción	Comentario	Resuelto
8	Ubicación del usuario en la aplicación		✓
9	Opción "Cambiar Usuario"	Se ha incluido esta opción en el menú de "Ajustes"	✓
10	Opción "Recordar Usuario"	Se ha incluido una casilla de verificación en el menú de inicio de sesión y la posibilidad de desactivar esta opción en el menú "Ajustes"	✓
11	Eliminar "Selección de Mapas"		✓

Tabla 5 - Resumen de los problemas detectados

8.2.2 Modificaciones Realizadas

En las siguientes capturas de pantalla se pueden ver algunas de las modificaciones realizadas a partir de los comentarios obtenidos por parte de los usuarios. Sólo se presentan las que han supuesto un cambio significativo en apariencia y/o funcionalidad.

- Se ha modernizado la imagen de fondo de manera que se muestra parte del nombre de la aplicación en la parte superior (visible en la pantalla de login) y la imagen de la Concha de Santiago en la parte inferior. Para mostrar que la aplicación está abierta a peregrinos de procedencias muy diversas, el interior de la concha se ha diseñado de manera que simule un mapamundi:

Ilustración 82 - Nuevo fondo de la aplicación

- En cuanto a la ubicación de los accesos directos, se puede apreciar la galería de la pantalla principal, que facilita la navegación del usuario dentro de la aplicación:

Ilustración 83 - Galería de Accesos Directos

- Los iconos escogidos según la tipología del recurso son los siguientes:

Punto de interés genérico:

Patrimonio:

Alojamiento:

Servicio:

Ocio:

Evento:

Ruta:

Foto:

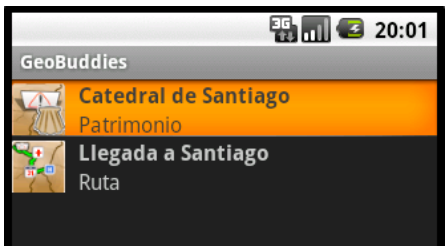


Ilustración 84 - Iconos de los Recursos

- Tal y como se sugirió en las pruebas de usuario, se han incluido descripciones de texto fijas en los iconos del panel de usuarios:

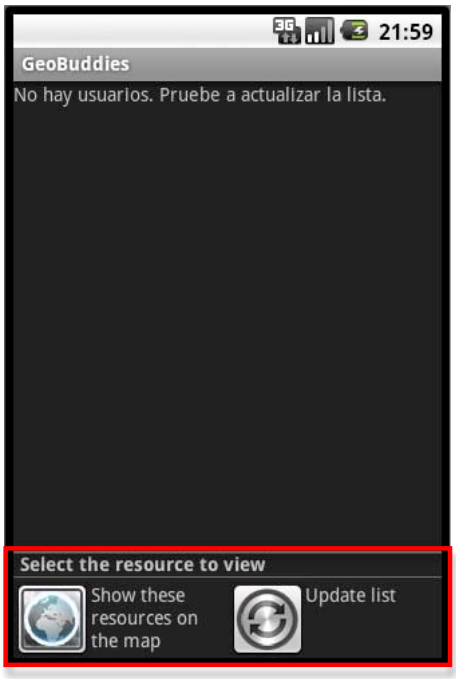


Ilustración 85 - Textos descriptivos en el panel de usuarios

- En el momento de la prueba de la aplicación por parte de los usuarios, no a todas las actividades de la misma se les había atribuido un título. Esto suponía que la navegación y usabilidad del sistema se complicaba un poco al no conocer el usuario, en todo momento, en qué lugar de la aplicación se encontraba.

  Se ha considerado importante este aspecto, de tal manera que, siempre que el usuario sale de la pantalla principal, se le informa de la tarea que se puede realizar desde la pantalla a la que ha accedido o lo que la información en ésta representa.

Se muestran algunos ejemplos:

- Tal y como se explicó en la sección anterior, se ha incluido la funcionalidad que permite a un usuario recordar sus datos de inicio de sesión, así como modificar el usuario asociado al terminal en cada momento:

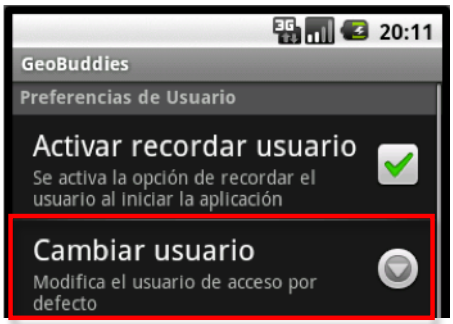


Ilustración 86 - Opción "Cambiar Usuario" en el menú de ajustes

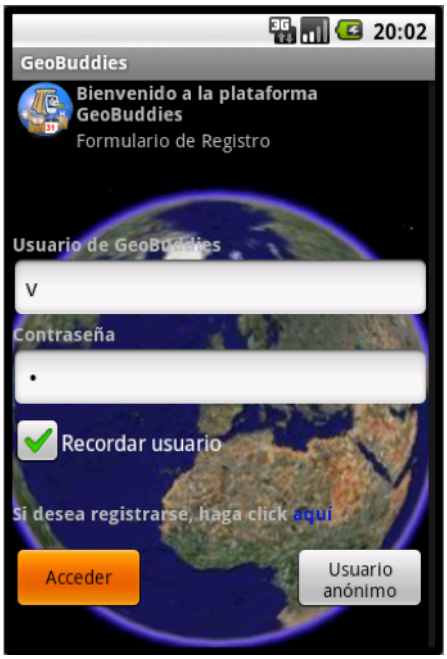


Ilustración 87 - Opción "Recordar Usuario" durante el login

- Ya que la aplicación siempre utiliza los mapas de Google, se procede a eliminar la opción que se daba en la sección de "Ajustes" de elegir los mapas con los que el usuario deseaba trabajar. Dicha opción estaba basada en la aplicación desarrollada previamente para Symbian, SO con el que Google Maps no está tan integrado.

Ilustración 88 – Previo a eliminar la funcionalidad

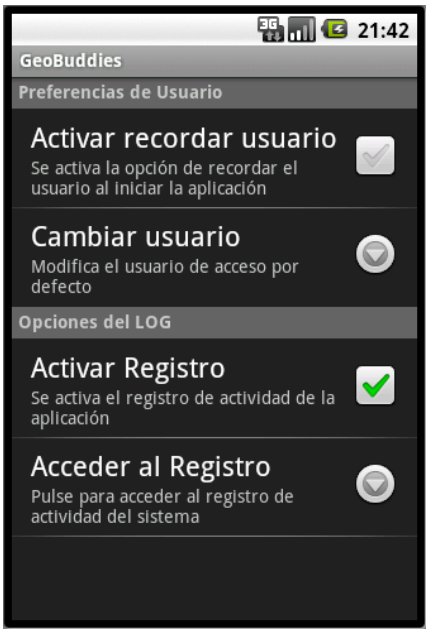


Ilustración 89 - Tras eliminar la funcionalidad

9 Conclusiones y Líneas Futuras

9.1 Lecciones Aprendidas

Gracias a la realización de este proyecto de fin de carrera he adquirido y reforzado conocimientos sobre diversas tecnologías necesarias para desarrollar aplicaciones móviles, a pesar de que, en este caso, se haya experimentado con aplicaciones para el sistema operativo Android exclusivamente.

Esta aplicación ha supuesto un alto trabajo de investigación del mercado existente de las tecnologías móviles, tanto en el ámbito de software como en el de hardware. Una de las dificultades evidentes que se presenta a la hora de desarrollar este tipo de sistemas es el del cambio constante de versiones de herramientas y librerías de desarrollo, lo que demuestra el gran número de avances que se realizan en este campo y el interés que existe en construir sistemas que dotan a las personas de un nivel de comunicación y accesibilidad a la información sorprendente.

En cuanto a las posibilidades que ofrece el desarrollo para Android, se ha podido comprobar que son muchas, y muy variadas en tipología, las aplicaciones que se pueden implementar. La aplicación desarrollada sólo es capaz de abarcar algunas de estas posibilidades pero, sin lugar a dudas, deja una clara idea de que se puede conseguir casi cualquier cosa en el marco de las comunicaciones.

No puedo, tampoco, dejar de comentar el hecho de que existen numerosos foros y una amplia documentación (ya desde el lanzamiento del primer terminal basado en Android) acerca del desarrollo para Android, claro indicio de las altas expectativas que ha levantado el surgimiento de este sistema operativo. Se ofrecen multitud de librerías y un conjunto de herramientas básico, vital para cualquier programador que quiera conocer lo que Android tiene que ofrecer de una manera muy simple.

Mis expectativas a la hora de decidirme por realizar este TFC eran las siguientes:

- Investigar el funcionamiento del desarrollo de aplicaciones sobre dispositivos móviles.

- Desarrollar una aplicación móvil que me sirviese de base para la implementación de futuras ideas y poder así contribuir en el avance de este tipo de tecnologías.

Con esta aplicación siento que he cumplido estos objetivos iniciales que me propuse, ya que me ha servido de base para poder experimentar con este joven sistema operativo para móviles, y me ha proporcionado los conocimientos y la familiarización necesaria para el desarrollo de posibles aplicaciones futuras.

9.2 Líneas Futuras

Tal y como se ha hecho ver en el apartado de pruebas de usuario, este proyecto presenta una serie de líneas futuras de investigación y desarrollo:

- Mejoras de interfaz
- Incorporación de nuevas funcionalidades

En el ámbito de las nuevas mejoras funcionales, se podría estudiar el manejo de otros medios multimedia (vídeos o grabaciones de voz) así como la inclusión del geoposicionamiento en el aspecto más social de la aplicación. Con esto último me refiero a la incorporación de la ubicación actual de "nuestros amigos", siempre y cuando éstos lo permitan, en las pantallas de "Mi Comunidad" y como posible capa en la vista de mapas (algo similar a lo que ofrece Google Latitude).

Por otro lado, se han presentado varias alternativas de sistemas operativos que se encuentran a la vanguardia en cuanto a desarrollo de aplicaciones. Éstos, claramente, suponen una competencia directa a los terminales con sistema operativo Android, por lo que, quizás, como ya se ha conseguido con esta aplicación, se podría analizar la viabilidad de portarla a otros sistemas operativos en auge.

En última instancia, destacar que, en el momento de finalización de este TFC se ha constatado el hecho de que la comunicación entre la aplicación Android y los Servicios Web mediante el protocolo SOAP (en su versión más ligera kSOAP) no ha sido completamente satisfactoria.

El problema radica en que la aplicación es capaz de obtener respuestas por parte del servidor al realizar consultas simples como, por ejemplo, la obtención de los datos de un usuario (envío únicamente del identificador en el mensaje de consulta). Sin embargo, cuando se intenta enviar un objeto complejo como, por ejemplo, un objeto "Usuario" con los datos de un usuario para poder registrarlo en el sistema, la aplicación devuelve un error de serialización:

```
08-12 23:19:54.710: INFO/RegistrarUsuario(298):  *** ejecutando llamada registrarUsuario
08-12 23:19:54.720: ERROR/RegistrarUsuario(298):  +++ Cannot serialize: geobuddies.service.Usuario@43e463e8
```

Ilustración 90 - Error de serialización de objetos complejos

Tras investigar el error e intentar buscar un solución, se ha llegado a la conclusión de que kSOAP2 no admite serialización. Algunas de las soluciones pensadas son:

- Des-serializar el objeto en el cliente para volverlo a serializar en el servidor (usando Marshal o similar).

  Se muestra un ejemplo de "marshaling" de argumentos, de manera que se puede apreciar la implementación de la interfaz Marshal para que el parser XML sepa cómo serializar y des-serializar objetos que se intentan pasar a través del servicio Web.

```
package Marshals;

import java.io.IOException;

import org.ksoap2.serialization.Marshal;
import org.ksoap2.serialization.PropertyInfo;
import org.ksoap2.serialization.SoapSerializationEnvelope;
import org.xmlpull.v1.XmlPullParser;
import org.xmlpull.v1.XmlPullParserException;
import org.xmlpull.v1.XmlSerializer;
/**
*
* @author Vladimir
* Used to marshal Doubles - crucial to serialization for SOAP
*/
public class MarshalDouble implements Marshal
{


    public Object readInstance(XmlPullParser parser, String namespace, String name,
            PropertyInfo expected) throws IOException, XmlPullParserException {

        return Double.parseDouble(parser.nextText());
    }


    public void register(SoapSerializationEnvelope cm) {
         cm.addMapping(cm.xsd, "double", Double.class, this);

    }


    public void writeInstance(XmlSerializer writer, Object obj) throws IOException {
           writer.text(obj.toString());
        }

}
```

Ilustración 91 - Ejemplo de marshaling de argumentos

- Otra opción consiste en seguir los principios REST (Representational State Transfer)[33], que no es más que una técnica de arquitectura software para sistemas hipermedia.

Sin embargo, el inconveniente de cualquiera de estas dos opciones es que exigen reescribir parte del código servidor, por lo que afectaría a la aplicación GeoBuddies ya existente.

Además, en el caso del uso de REST, la comunidad Android no proporciona muchas ayudas, aunque es cierto que cada vez más, por lo requiere una investigación más exhaustiva en el tema con el consiguiente acarreo de tiempo. Por tanto, se propone la búsqueda de la solución más adecuada y su consecuente implementación para una futura línea de investigación.

[33] http://rest.blueoxen.net/cgi-bin/wiki.pl

Anexo A. Ejemplos de Código de Componentes Android

De manera que se facilitase la lectura de la memoria, se ha decidido excluir los ejemplos de código de los componentes principales de Android del cuerpo principal de ésta. Por este motivo, se incluye a continuación una recopilación que ayude al lector a comprender el modo de funcionamiento de dichos componentes.

A.1 Servicios

Se crean y destruyen mediante un intent:

```
/* Se crea un intent que indica la clase que implementa el servicio y se
inicia mediante el método startService */
Intent svc = new Intent(this, MyService.class);
startService(svc);
…
/* Para detener el servicio, se procede del mismo modo, sólo que
invocando el método stopService */
Intent svc = new Intent(this, MyService.class);
stopService(svc);
```

Un servicio implementa una clase que lo gestiona y que extiende a la clase Service para que pueda ejecutarse correctamente. Habrá una serie de hilos ejecutando en segundo plano:

```
/* Se crea un intent que indica la clase que implementa el servicio y se
inicia mediante el método startService */
public class MyService extends Service {

	/* Métodos similares a los de cualquier Activity */
	@Override
	public void onCreate() {
		super.onCreate();
		/* Método que realiza toda la actividad */
		startService();
	}

	@Override
	public void onDestroy() {
		super.onDestroy();
		shutdownService();
	}

	private void _startService() {
	/* Una práctica habitual es la de crear un temporizador que lance un
	hilo que realice una tarea */
	timer.scheduleAtFixedRate(new TimerTask() {
		public void run() {
```

```
                Thread cThd = new Thread(new claseTarea());
                cThd.start();
            }
            }, 0, UPDATE_INTERVAL);
        }

        private void _shutdownService() {
            /* Se destruye el temporizador */
            if (timer ¡= null)
                timer.cancel();
        }

}
```

A.2 Receptores de difusión

Existen tres maneras distintas de implementar receptores de difusión (*listeners*) en Android, cada una de las cuales presenta una serie de ventajas e inconvenientes:

- Implementación embebida en la invocación
- Implementando una interfaz de *listener* específica
- Mediante variables

En el primer caso, se crea un *listener* anónimo en el que se definen y pasan las funciones en el mismo paso de creación:

```
btnLogin.setOnClickListener(new OnClickListener() {
        public void onClick(View v) {
            /* Acción a realizar al hacer click en la vista */
        }
});
```

Las ventajas de esta primera aproximación son que la implementación es breve, simple y ordenada. Sin embargo, es muy inflexible ya que no permite reutilización (recordemos que es anónimo) y que puede ser más difícil de mantener.

El enfoque de implementar una interfaz de *listener* específica es más reusable aunque puede llevar a código más desorganizado (con muchos bloques *if – elseif – else* si las acciones a ejecutar son muy dispares):

```
public class EjemploLogin extends Activity implements OnClickListener {

        /*Se invoca cuando se crea la actividad por primera vez */
        @Override
        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
```

```
            ...

            /* Se añade el lstener para el click en el botón */
            btnLogin.setOnClickListener(this);
            btnCancel.setOnClickListener(this);
      }

      @Override
      public void onClick(View v) {
            if(v==btnLogin) {
                  /* Verificación de login */
                  ...
            } else if(v==btnCancel) {
                  /* Cierre de la aplicación */
                  finish();
            }
      }
}
```

Mediante la utilización de variables, se consigue que la implementación no forme parte de nuestra clase, sino que se tenga una referencia al *listener* en una variable:

```
OnClickListener lAcceso = new OnClickListener() {
      public void onClick(View v) {
            /* Acción a realizar al hacer click en la vista */
            ...
      }
};
```

Se trata, al igual que en el primer caso, de un *listener* anónimo pero reutilizable, ya que el usuario dispone de una referencia al mismo.

Además, Android implementa una serie de *listeners* en forma de métodos que pueden invocarse desde las actividades:

```
/* Método que se ejecuta al crearse un menú en la actividad */
@Override
public boolean onCreateOptionsMenu(Menu menu) {
      …
}

/* Método que se ejecuta al seleccionar un elemento de un menú */
@Override
public boolean onMenuItemSelected(int featureId, MenuItem item) {
      …
}

/* Método que se ejecuta al retornar a la actividad en la que se implementa
(tras haber invocado a otra actividad a través de startActivityForResult) */
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data){
      …
}
```

A.3 Proveedores de contenidos

Por lo general, el sistema operativo Android fuerza a que los datos de una aplicación estén fuertemente ligados a la misma, es decir, se asegura de que una aplicación no pueda leer o escribir datos entre paquetes.

Sin embargo, existirán aplicaciones que quieran compartir datos y permitir que otras aplicaciones externas puedan leer y escribir datos en su base de datos. Un claro ejemplo son los datos de nuestros contactos. Si Android no proveyese una manera de compartir esta información entre aplicaciones, se forzaría al usuario a mantener una base de datos diferente para cada aplicación específica, y no habría ningún tipo de vínculo entre unos datos y otros.

Para permitir esto, existe la API de proveedor de contenidos, a través de la cual se permite a cada aplicación cliente a que solicite la información que necesite al propio sistema operativo, utilizando un mecanismo de identificación uniforme de recursos (de las siglas en inglés URI).

La API de proveedor de contenidos dotará a la aplicación acceso total al contenido, por lo que ésta podrá:

- Crear nuevos registros
- Recuperar uno, todos, o un conjunto limitado de registros
- Actualizar registros
- Borrar registros, siempre y cuando esto esté permitido

La interfaz común que implementan todos los proveedores de contenidos se emplea a través de objetos *ContentResolver*. Estos objetos se obtienen invocando al método *getContentResolver()* desde la implementación de una actividad o cualquier otro componente:

```
ContentResolver cr = getContentResolver();
```

Seguidamente vemos un ejemplo de extracción e inserción en el proveedor de contenidos de los contactos:

```
/* Extracción de un dato */
/* 1º.- Se obtiene la URI para la tabla People */
Uri mContacts = People.CONTENT_URI;

/* Se extrae el contenido como si fuese una base de datos */
Cursor mCursor = managedQuery(mContacts,
                              projection,
                              null,
                              null,
                              People.NAME + " ASC");

...

/* Inserción de un dato */
ContentValues values = new ContentValues();

/* Se añade el valor indicando su tipo (el del proveedor de contenidos destino
-> Contacts) */
values.put(Contacts.People.NAME, "Un Nombre");

/* Se inserta el dato */
Uri uri = getContentResolver().insert(Contacts.People.CONTENT_URI,
                                      values);
```

A.4 *Intents*

Para pasar de una actividad a otra, Android dispone de la clase *Intent*. Existen dos invocaciones distintas para pasar de una actividad a otra.

La primera es el paso a la espera de una confirmación/respuesta por parte de la actividad invocada:

```
/* Se crea un Intent indicando la clase a ejecutar */
Intent i = new Intent(this, Login.class);

/* Se invoca el método que inicia la nueva actividad */
this.startActivityForResult(i, codigoRespuesta);

...

/* En la actividad destino, una vez finalizada la tarea y antes de indicar el
regreso a la actividad origen, se indica el resultado de la operación */
setResult(RESULT_OK);
finish();

...
/* De vuelta e la actividad origen, se procede a evaluar el resultado de la
operación */
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data){
      ...
}
```

La segunda, es la invocación a la nueva actividad sin esperar ningún resultado:

```
/* Se crea un Intent indicando la clase a ejecutar */
Intent i = new Intent(this, Login.class);
/* Se invoca el método que inicia la nueva actividad */
this.startActivity(i);
finish();
```

Anexo B. Android Manifest

B.1 ¿Qué es el fichero *AndroidManifest*?

Como ya se ha mencionado anteriormente, el fichero *AndroidManifest* es el fichero de control que indica al sistema central lo que debe hacer con todos los componentes que conforman la aplicación. Es especialmente útil para indicarle qué tipo de datos puede manejar o de qué permisos dispone la aplicación en cuanto a ejecución (si tiene acceso a la red, al dispositivo GPS, etc.).

Su inclusión en el directorio raíz del proyecto es obligatoria, ya que sin este fichero, no se podrá compilar ni ejecutar.

B.2 Ejemplo básico

A continuación se incluye, a modo de ejemplo, un extracto del manifiesto de la aplicación desarrollada:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.geobuddies"
      android:versionCode="1"
      android:versionName="1.0">

    <supports-screens android:smallScreens="true"
          android:normalScreens="false"
          android:largeScreens="false"
          android:anyDensity="false" />

    <uses-permission android:name="android.permission.LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_MOCK_LOCATION" />
    …
    <uses-permission android:name="android.permission.ACCESS_GPS" />
    <uses-permission android:name="android.permission.CAMERA" />
    …
    <uses-permission android:name="android.permission.INTERNET" />
    …

    <application android:icon="@drawable/icon" android:label="@string/app_name">
          <uses-library android:name="com.google.android.maps" />

        <activity android:name=".GeoBuddies"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".FormLogin" />
```

```
    <activity android:name=".FormRegistro" />
    <activity android:name=".FormCambioUsuario">
      <intent-filter>
            <action android:name="com.geobuddies.CHG_USER" />
            <category android:name="android.intent.category.DEFAULT" />
      </intent-filter>
    </activity>
    <activity android:name=".FormPrincipal" />
    <activity android:name=".FormUsuarios" />
    <activity android:name=".FormMostrarDetallesUsuario" />
    <activity android:name=".BuscarAmigos" />
    <activity android:name=".FormMostrarRecursosUsuario" />
    <activity android:name=".FormPreferencias" />
    …
</application>

<uses-sdk android:minSdkVersion="3" />

</manifest>
```

Como se puede apreciar, el manifiesto incluye, en primer lugar, una serie de permisos de los que debe disponer la aplicación para que se ejecute normalmente. Esta lista de permisos, y recursos que requiere la aplicación, es la que se le presenta al usuario final antes de instalar la aplicación para que dé su consentimiento. Toda funcionalidad protegida[34], a la que se quiera tener acceso desde el sistema desarrollado, deberá incluirse en el manifiesto mediante el tag `<uses-permission>`.

Tras establecer los permisos, se definen todas las actividades disponibles en el paquete de la aplicación desarrollada. Todas deben estar bajo el tag `<application>` y se declaran mediante el tag `<activity>`. Si una actividad no se ha declarado en este momento, la aplicación no podrá lanzarla y el sistema devolverá un error de acceso a una actividad no existente.

Cualquier paquete que se presente como aplicación de alo nivel deberá incluir una actividad que soporte la acción *main* y la categoría *launcher*, ya que esta actividad es la que se lanzará al arrancar el sistema:

```
      <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
```

[34] La lista completa de los permisos que se pueden establecer para una aplicación Android se puede encontrar en: http://code.google.com/intl/es-ES/android/reference/android/Manifest.permission.html

Anexo C. Uso del DDMS

C.1 ¿Qué es DDMS?

Para facilitar la tarea de depuración del desarrollador, Android se distribuye con una herramienta de depuración llamada *Dalvik Debug Monitor Service*. Esta herramienta proporciona, entre otros, los siguientes servicios:

- Redirección de puertos
- Captura de pantalla del dispositivo
- Información sobre la pila y los hilos de ejecución del dispositivo
- Información sobre el estado de los procesos
- Simulación de llamadas y envíos de mensajes de texto
- Simulación de recepción datos de localización geográfica

C.2 Ubicación y modo de uso

DDMS se encuentra en el directorio de herramientas ("tools/") de SDK. Esta herramienta funcionará tanto con el emulador como con un dispositivo real conectado. Si ambos están conectados y ejecutándose simultáneamente, *DDMS* cogerá el emulador por defecto.

DDMS actúa como intermediario para conectar el entorno redesarrollo a las aplicaciones que se están ejecutando en el dispositivo/emulador. En Android, y como ya se ha mencionado anteriormente en este trabajo, cada aplicación ejecuta en un proceso propio, cada uno de los cuales hospeda su propia máquina virtual, por lo que cada proceso escucha a un depurador en un puerto distinto.

Cuando arranca, *DDMS* se conecta al adb (de las siglas en inglés, "Android Debug Bridge") y ejecuta un servicio de monitorización de dispositivos entre ambos, que notificará al *DDMS* en el momento en que se conecte o desconecte un dispositivo. Al conectar un dispositivo, se crea un servicio de monitorización de máquina virtual entre el adb y el *DDMS*, que notificará a este último cuando ua máquina virtual se arranca o para en el emulador. Una vez que hay una máquina virtual en ejecución, el *DDMS* obtiene el identificador del proceso de la máquina virtual, vía el adb, y abre una

conexión al depurador de dicha máquina virtual. De esta forma queda establecido un canal de comunicación entre el *DDMS* y la máquina virtual.

El monitor de depuración abre un puerto para cada máquina virtual en el dispositivo, en el que escuchará a la espera de un depurador. Por defecto, para la primera máquina virtual, el *DDMS* escucha en el puerto 8600. Si se ejecutasen otras aplicaciones simultáneamente, los puertos de escucha serían el 8601, 8602, y así sucesivamente.

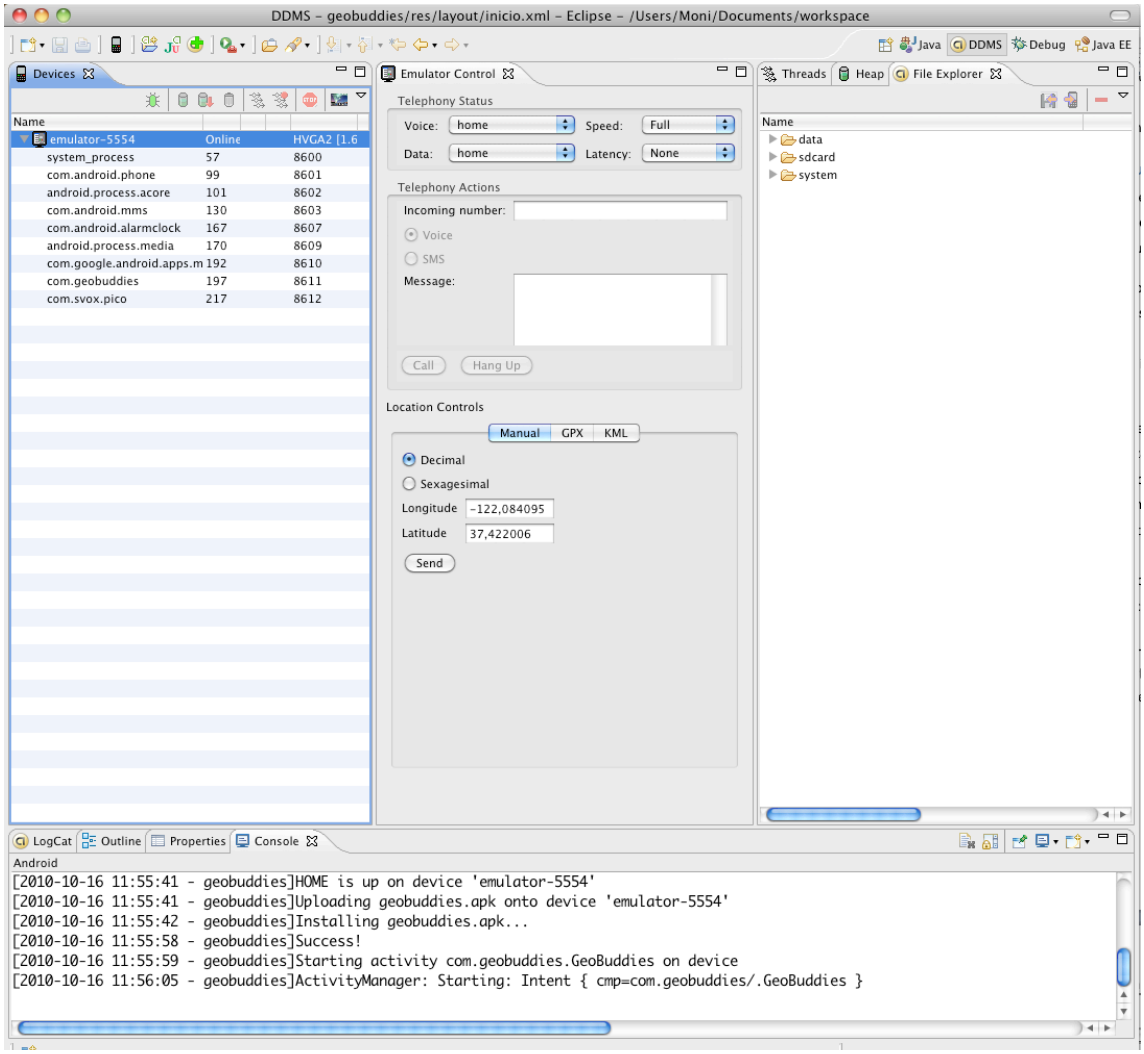


Ilustración 92 - Dalvik Debug Monitor Service

C.2.1 Panel Izquierdo

El panel izquierdo del monitor de depuración muestra cada uno de los emuladores/dispositivos activos, con una lista de todas las máquinas virtudes ejecutándose en cada uno. Éstas se identifican por e nombre del paquete de la aplicación que alberga.

En esta lista se podrán seleccionar las actividades que se quieran depurar dentro de cada máquina virtual. Junto a cada máquina virtual de la lista de encuentra un número de puerto ª(columna derecha) a través del cual, si conectamos el depurador al mismo, estaremos conectados a dicha máquina virtual en el dispositivo mostrado. Sn embargo, *DDMS* nos facilita la tarea, ya que únicamente será necesario conectarse al puerto 8700, ya que *DDMS* redirige todo el tráfico de este puerto a la máquina virtual seleccionada en cada momento. Esto permite que no sea necesario reconfigurar el puerto del depurador cada vez que se quiera saltar de una máquina virtual a otra. Nótese que, al seleccionar una máquina virtual de la lista, el puerto mostrado incluye el 8700:

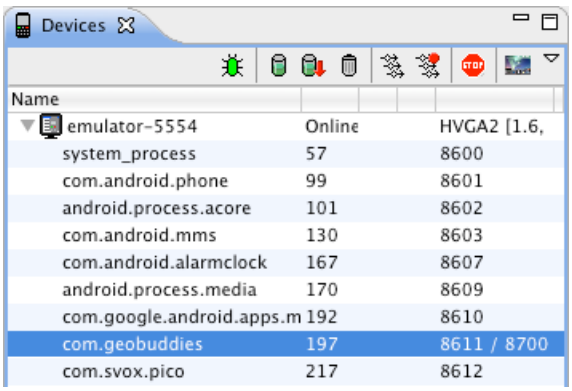


Ilustración 93 - Selección de una máquina virtual

C.2.2 Panel Derecho

En el lado derecho, el monitor de depuración proporciona una serie de pestañas que muestran información general del emulador seleccionado. Sin duda, la vista más útil es la que se muestra en la Ilustración 83, el explorador de archivos.

Esta pestaña permite ver el sistema de ficheros del dispositivo que estamos depurando así como realizar gestiones básicas sobre el mismo, como, por ejemplo, eliminar e insertar ficheros. Asimismo, si se ha montado una imagen de tarjeta SD en el emulador, se podrá acceder a los archivos almacenados en ella desde este mismo panel.

C.2.3 Panel Central

Finalmente, en el centro del monitor se ofrecen al desarrollador una serie de herramientas que permiten emular ciertas funciones:

- Llamadas
- Envío de SMS
- Controles de localización GPS

El que se ha empleado para el desarrollo de este proyecto ha sido el de localización, puesto que permite emular la recepción de coordenadas de posicionamiento GPS en el dispositivo. De esta forma ha sido posible probar y depurar las funcionalidades sobre el mapa de Google.

El funcionamiento de esta herramienta es muy sencillo, ya que, una vez que nuestra aplicación se encuentre a la espera de recepción de coordenadas (por ejemplo, al entrar en la vista del mapa de GeoBuddies), se deberán insertar unas coordenadas (en el formato que mejor nos convenga) en las casillas del *DDMS* y pulsar el botón "Send". Es en este momento cuando el dispositivo recibirá la información, y el cursor del mapa navegará hasta la posición introducida en el *DDMS*.

Anexo D. Obtención de una clave para la API de Google Maps

D.1 ¿Por qué es necesaria una clave de registro?

Para integrar los mapas de Google con nuestra aplicación, es necesario usar la clase *MapView* de la librería externa *Maps*. Esta clase permite la descarga, renderizado y copiado en caché de las cuadrículas del mapa, así como proporciona una variedad de opciones de visualización y controles.

Es necesario realizar un registro con el servicio de mapas de Google y aceptar los términos y condiciones del mismo porque *MapView* te da acceso a los datos de Google Maps. Sin un registro previo, la vista de mapa de la aplicación no será capaz de obtener datos de Google Maps, siendo necesario tanto si se está desarrollando la aplicación en un emulador como si se va a desplegar en terminales reales.

D.2 Cómo obtener una clave de registro

El registro para obtener una clave de Google Maps es gratuito y requiere dos pasos:

1. Se debe registrar la huella MD5 del certificado que se use para firmar la aplicación. El servicio de registro proporcionará al desarrollador con una clave asociada a dicho certificado.
2. Una vez se ha obtenido la clave, se debe añadir una referencia a cada *MapView*, ya sea en el fichero XML de la clase o como código en el fuente de dicha clase. Se puede emplear la misma clave en cualquier aplicación Android siempre que dicha aplicación esté firmada con el certificado cuya huella se registró en el servicio.

En este proyecto se ha firmado la aplicación en modo depuración (el SDK dispone de un certificado de depuración).

Para generar la huella MD5 de este certificado, lo primero que hay que hacer es localizar el fichero "debug keystore". En Mac OS y Linux se encuentra en la siguiente ruta: */Users/<user>/.android/debug.keystore* y en Windows XP en: *c:\Documents and Settings*\<*user*>\LocalSettings\ApplicationData\Android\debug*.keystore*.

Una vez localizado, se usará la herramienta *keytool* para obtener la firma MD5.

```
moniMac:~ Moni$ pwd
/Users/Moni
moniMac:~ Moni$ keytool -list -alias androiddebugkey -keystore /Users/Moni/.android/d
ebug.keystore -storepass android -keypass android
androiddebugkey, 22-ene-2010, PrivateKeyEntry,
Huella digital de certificado (MD5): 52:E3:93:33:BC:27:B6:00:46:8C:79:3A:FE:6C:67:72
```

Ilustración 94 - Obtención de la firma MD5 del certificado de depuración

Ya con la firma MD5, se puede acceder al servicio de registro en la siguiente URL: http://code.google.com/intl/es-ES/android/maps-api-signup.html y se registra. (El desarrollador debe disponer de una cuenta de Google para poder acceder).

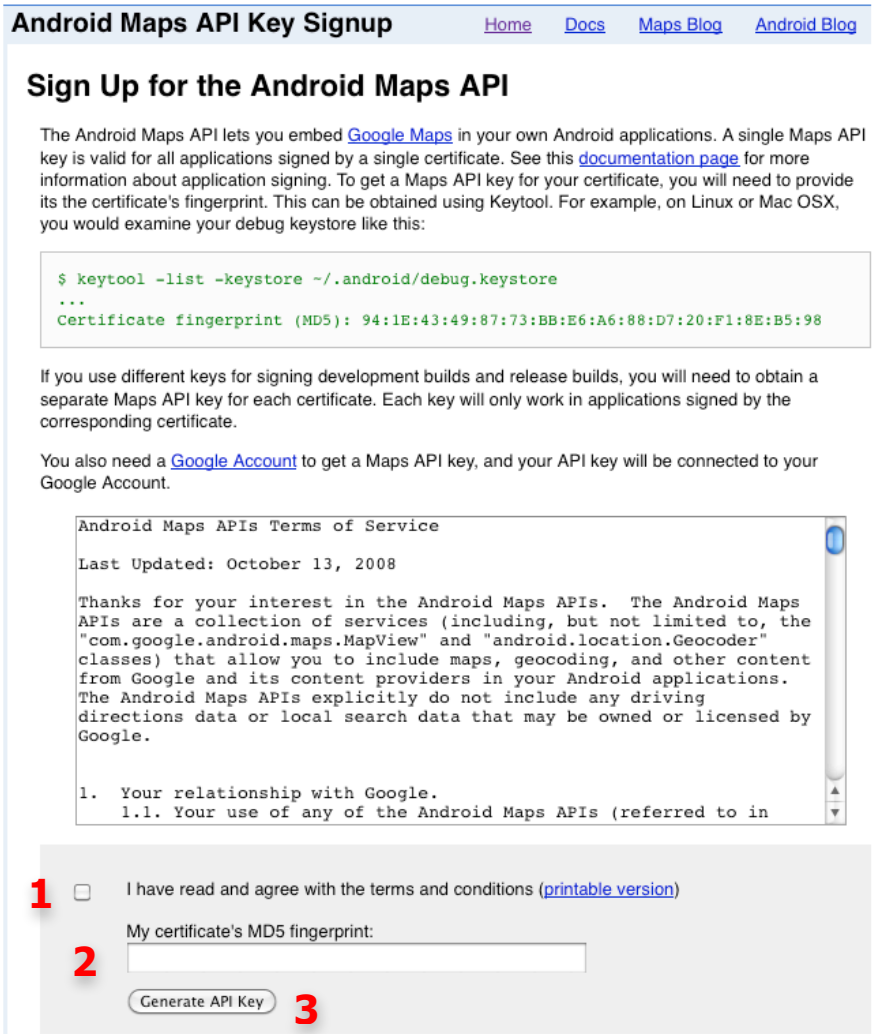


Ilustración 95 - Registro de la firma MD5

En este momento, se pasaría a la segunda parte mencionada anteriormente para añadir la firma a las clases *MapView* de la apicación.

```
<com.geobuddies.mapas.MyMapView
    android:id="@+id/mapview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:enabled="true"
    android:clickable="true"
    android:apiKey="0lucmQFHGBKPtsq-sTj2rtAWh9joUSOV5uvNdOw" />
```

Ilustración 96 - Referencia XML a la clave obtenida para Google Maps

Anexo E. Documentos de la Evaluación del Sistema

E.1 Instrucciones del Test

Gracias por tu ayuda en la evaluación de la usabilidad de la aplicación GeoBuddies sobre Android.

Soy alumna de la Facultad de Informática de la UPM y voy a llevar a cabo contigo el test de usabilidad. Puedes hacerme cualquier pregunta sobre el test y sobre lo que se te pide en cualquier momento.

La aplicación que se va a evaluar está basada en una previa ya existente y tiene el fin de ejecutarse sobre dispositivos basados en Android como el que se te va a facilitar para las pruebas. La idea de la aplicación es poder ofrecer a sus usuarios (peregrinos del Camino de Santiago) información en tiempo real sobre lugares de interés y rutas que se encuentren cercanas a su ubicación actual. Asimismo, se les ofrece la posibilidad de dar de alta nuevos puntos de interés (recursos) para compartirlos con la comunidad virtual existente.

Te voy a pedir que imagines que estás en distintas situaciones en las que necesitas llevar a cabo alguna de las funcionalidades que permite la aplicación, y que intentes utilizarla para satisfacer tus necesidades. El objetivo de la evaluación es identificar los problemas y fallos en la estructura de la información y en los nombres de los enlaces, así como obtener sugerencias sobre posibles mejoras.

Querría que tuvieras claro que no se te estará evaluando a ti, sino a la aplicación móvil. Siempre que tengas dificultad para encontrar algo será una indicación de un problema de usabilidad y todos los problemas que me ayudes a descubrir me permitirán mejorar la usabilidad del sistema.

Durante las acciones que vas a realizar vamos a medir el tiempo que tardas en realizarlas como indicador de la usabilidad de la aplicaciónn. Tu nombre no se asociará con ninguno de los datos recogidos en la evaluación. Los informes de la evaluación se redactarán de tal forma que los participantes en los test no puedan ser identificados individualmente.

Una vez hayas llevado a cabo las tareas que se te van a pedir, te daré un cuestionario para poder recoger tu opinión sobre el sistema.

E.2 Escenarios Propuestos

E.2.1 Registrarse en la Aplicación

Suponte que eres un peregrino del Camino de Santiago y que dispones de un teléfono inteligente con sistema operativo Android. En tu terminal se ha instalado una nueva aplicación que deseas poder usar durante tu peregrinación para poder obtener información acerca de los lugares y rutas de interés cerca de tu ubicación.

Para empezar a usarla, primero debes registrarse en la comunidad virtual. Por favor, arranque la aplicación (el acceso directo se encuentra en el escritorio principal) y lleve a cabo esta acción.

E.2.2 Iniciar Sesión en la Aplicación y Recordarla

Suponte que vas a utilizar muy a menudo la aplicación, por lo que quieres asociar "permanentemente" tus datos de inicio de sesión con el dispositivo que estás utilizando.

E.2.3 Cambiar Usuario de Inicio Asociado

Durante tu viaje te encuentras con un viejo conocido que, casualmente, dispone de un terminal basado en Android y es miembro de la comunidad virtual de GeoBuddies. Ha tenido un problema y su dispositivo está sin batería. Como le prometió a un amigo que subiría la ruta realizada ese día a la comunidad virtual para que tuviese acceso a ella, le ofreces tu terminal. Deberás cambiar el usuario de acceso asociado a la aplicación.

E.2.4 Agregar Recurso

Suponte que en el día de hoy has realizado un tramo de ruta muy interesante, ya que los pueblos por los que has pasado te han parecido muy pintorescos y dignos de visitar. Has grabado dicha ruta en tu terminal con un programa destinado a ello y ahora quieres compartirla con tus amigos de GeoBuddies.

La ruta se encuentra en la siguiente ubicación de tu dispositivo: /sdcard/rutas/ruta.kml

E.2.5 Visualizar Recursos de un Amigo

Has conocido a otro peregrino usuario (Usuario Amigo) de la aplicación durante el camino. Te dispones a consultar los recursos que tiene compartidos, puesto que te ha comentado que muy cerca de donde te encuentras ha visitado recientemente un "edificio de gran interés". Deberás buscar el edificio entre los recursos de tu amigo y consultar los detalles.

Nota.- Un usuario sólo es amigo tuyo si lo agregas primero a tu comunidad.

E.2.6 Consultar recursos sobre el mapa

Suponte que quieres consultar los recursos que has ido agregando y su ubicación gráficamente. Accede a la vista de mapa y navega por él (haciendo el zoom necesario) para visualizarlos.

E.2.7 Resto de Funcionalidad

Por último, te pido que, ahora que estás familiarizado/a con la aplicación, te dispongas a investigarla durante unos 5 minutos para que puedas comentarme mejoras o funcionalidad extra que crees que sería útil.

E.3 Plantilla de Anotaciones de la Pruebas

Participante:

Perfil:

Fecha:

- *Suponte que eres un peregrino del Camino de Santiago y que dispones de un teléfono inteligente con sistema operativo Android. En tu terminal se ha instalado una nueva aplicación que deseas poder usar durante tu peregrinación para poder obtener información acerca de los lugares y rutas de interés cerca de tu ubicación. Para empezar a usarla, primero debe registrarse en la comunidad virtual. Por favor, arranque la aplicación (el acceso directo se encuentra en el escritorio principal) y lleve a cabo esta acción.*
  No Clicks:
  Tiempo:
  Errores:

  Comentarios:

- *Suponte que vas a utilizar muy a menudo la aplicación, por lo que quieres asociar "permanentemente" tus datos de inicio de sesión con el dispositivo que estás utilizando.*
  No Clicks:
  Tiempo:
  Errores:

  Comentarios:

- *Durante tu viaje te encuentras con un viejo conocido que, casualmente, dispone de un terminal basado en Android y es miembro de la comunidad virtual de GeoBuddies. Ha tenido un problema y su dispositivo está sin batería. Como le prometió a un amigo que subiría la ruta realizada ese día a la comunidad virtual para que tuviese acceso a ella, le ofreces tu terminal.*
  Deberás cambiar el usuario de acceso asociado a la aplicación.
  No Clicks:
  Tiempo:
  Errores:

  Comentarios:

- *Suponte que en el día de hoy has realizado un tramo de ruta muy interesante, ya que los pueblos por los que has pasado te han parecido muy pintorescos y dignos de visitar. Has grabado dicha ruta en tu terminal con un programa destinado a ello y ahora quieres compartirla con tus amigos de GeoBuddies.*
  La ruta se encuentra en la siguiente ubicación de tu dispositivo: /sdcard/rutas/ruta.kml
  No Clicks:
  Tiempo:
  Errores:

  Comentarios:

- *Has conocido a otro peregrino usuario (Usuario Amigo) de la aplicación durante el camino. Te dispones a consultar los recursos que tiene compartidos, puesto que te ha comentado que muy cerca de donde te encuentras ha visitado recientemente un "edificio de gran interés". Deberás buscar el edificio entre los recursos de tu amigo y consultar los detalles.*
  Nota.- Un usuario sólo es amigo tuyo si lo agregas primero a tu comunidad.
  No Clicks:
  Tiempo:
  Errores:

  Comentarios:

- *Suponte que quieres consultar los recursos que has ido agregando y su ubicación gráficamente. Accede a la vista de mapa y navega por él (haciendo el zoom necesario) para visualizarlos.*
  No Clicks:
  Tiempo:
  Errores:

  Comentarios:

E.4 Cuestionario de Impresiones

Participante:

Perfil:

Fecha:

- ¿Cuáles son los principales problemas que has encontrado al acceder a esta aplicación?
- ¿Cuáles son las características (positivas) más destacables para ti?
- ¿Cuál es la parte de la aplicación con la que has tenido más problemas?
- ¿Cuál es la parte del sistema que crees que es la más oscura o difícil de entender?
- ¿Puedes describir tu experiencia general al acceder a la aplicación?
- ¿Piensas que hay algún tipo de acción vital que se ha pasado por alto?

E.5 Cuestionario de Satisfacción

Participante:

Fecha:

Por favor, rellena el siguiente cuestionario sobre tu impresión general de la aplicación.

1. Forma en la que la aplicación permite realizar las tareas solicitadas
Frustrante 1 2 3 4 5 Fácil

2. La navegación a través de la aplicación resulta
Confusa 1 2 3 4 5 Muy Clara

3. Apariencia general de la aplicación
Mala 1 2 3 4 5 Agradable

4. Estructura y organización de la aplicación
Confusa 1 2 3 4 5 Muy Clara

5. ¿Te han parecido claros y representativos los nombres de los botones y descripciones de la aplicación?
Confusos 1 2 3 4 5 Muy Claros

6. En general, ¿te fue fácil encontrar la información que buscabas?
Muy difícil 1 2 3 4 5 Muy Fácil

7. ¿Crees que la aplicaciónn es adecuada a las necesidades de distintos tipos de usuarios?
Nada adecuada 1 2 3 4 5 Muy adecuada

8. Facilidad de uso de la aplicación, en general
Confusa 1 2 3 4 5 Muy Clara

Bibliografía

Principios de la Web 2.0

Satyajeet Singh. *Web 2.0*
http://gonzbuk.com/2007/11/06/publicidad-internet-vigo-galicia-adwords-ourense-pontevedra-web-20-web-10/

Christian Van Der Henst S. *¿Qué es la Web 2.0?*
http://www.maestrosdelweb.com/editorial/web2/

Fundación Orange. *Mapa Visual de la Web 2.0*
http://internality.com/web20/

Wikipedia. *Web 2.0*
http://es.wikipedia.org/wiki/Web_2.0

Entornos Móviles

iOS Reference Library. *iOS Technology Overview*
http://developer.apple.com/library/ios/#documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/IPhoneOSOverview/IPhoneOSOverview.html

Android Developers. *What is Android?*
http://developer.android.com/intl/fr/guide/basics/what-is-android.html

Lee Graham. The NPD Group. *Android Shakes Up U.S. Smartphone Market*
http://www.npd.com/press/releases/press_100510.html

Varios

Berners-Lee. *URI Generic Syntax*
http://tools.ietf.org/html/rfc3986

Javier Pérez Pacheco. *TableDB*
http://android.javielinux.com/tabledb.php

Foros de Android

http://www.android-spa.com/index.php

http://developer.android.com/intl/fr/index.html